TRABAJO FIN DE GRADO

Estudio de una arquitectura concreta para la gamificación de aplicaciones, propuestas de mejora y diseño teórico de una arquitectura genérica.

GRADO EN INGENIERÍA DE SISTEMAS AUDIOVISUALES

Tutor: ESTÉVEZ AYRES, IRIA MANUELA

Alumno: ORTIZ HERNÁNDEZ, DANIEL

Universidad Carlos III de Madrid
www.uc3m.es

Índice

Capítulo 1: Introducción

1.1 Contexto

En el marco económico actual, en el que las empresas pretenden generar y mantener cada vez más clientes fieles a sus marcas, intentando evitar en la medida de lo posible el poco compromiso de éstos y la predisposición que se observa por parte de los usuarios a los cambios de firma, parecen necesarias nuevas técnicas que aumenten la atracción duradera hacia los productos o servicios que se ofrecen.

Por otro lado, existen gran cantidad de aplicaciones en entornos empresariales o educativos que tienen un uso muy escaso pese a su posible gran utilidad, causando en consecuencia pérdidas económicas que podrían ser evitadas de seguir una estrategia de enganche[1] que convenciera y motivara a los usuarios para no abandonarlas.

1.2 Estructura de la memoria

La presente memoria pretende desarrollar de forma ordenada tanto el aprendizaje de la técnica de la gamificación como las posibles soluciones

[1] Del inglés "*Engagement*"

comerciales que podrían desarrollarse para su correcta aplicación, estructurado como se explica a continuación

Empezando por la teoría, en el Capítulo 2 se explicarán los conceptos necesarios para entender la gamificación como técnica. Este capítulo responderá a preguntas tales como ¿Qué es la gamificación? ¿Cuál es su importancia en la actualidad y a qué se debe? ¿De dónde surgió la idea de aplicarla?. Además, a partir de un contexto histórico que pretende dar explicación al origen de la gamificación, se verá qué tipo de elementos son los que la forman y los que tienen mayor éxito en su aplicación, para así establecer desde un primer momento la línea de trabajo para la propuesta técnica.

Una vez conocida la teoría, en el capítulo 3 se hará un estudio de los requisitos que debería tener un buen sistema de gamificación a partir tanto de casos de estudio en los que se precise una solución gamificada como del aprendizaje de las arquitecturas ya estudiadas.

En el siguiente capítulo, se estudiarán las soluciones comerciales que actualmente se pueden encontrar en el mercado como arquitecturas de gamificación. En concreto, el estudio se centrará en una de las más accesibles en cuanto a código, para que a partir de los requisitos establecidos en el capítulo 3, se puedan proponer posibles mejoras en la implementación.

A partir de ahí, y usando los requisitos del capítulo 3 a modo de manual, se propondrá una solución técnica teórica de una arquitectura de gamificación que cumpla los objetivos propuestos, además de exponer el presupuesto teórico que supondría llevar a cabo el presente proyecto.

Por último, se presentarán las conclusiones que se pueden extraer después del trabajo realizado y se marcarán las líneas de trabajo futuro.

Capítulo 2: Gamificación

2.1 Qué es la Gamificación

"Gamificación" es el empleo de mecánicas y dinámicas de juego para adquirir hábitos y alcanzar objetivos [1].

Es el uso de elementos y diseños de los juegos en ámbitos que no están relacionados con el juego, por ejemplo, usar barras de progreso en páginas de búsqueda de empleo, trofeos en intranets de empresas, clasificaciones de puntuación en participación en los foros o la wiki de una asignatura, etc. Es decir, la gamificación se basa generar una motivación en actividades que por definición no tienen la obligación de ser entretenidas, con la intención de hacerlas más divertidas.

La gamificación no deja de ser algo que conocemos desde hace mucho tiempo, y cuyos **elementos** son los que hemos usado cuando jugamos:

- **Puntos**, por haber matado muchos zombies,
- **Niveles** en el Candy crush o en el mario bros.
- **Recursos** en el monopoly
- **Desafíos** (quests) como las contrarrelojes.
- **Avatares**, perfiles de habbo o los sims
- **Red social** para saber si voy ganando a mis amigos
- **Progreso** para mejorar mi personaje

- **Premios** como medallas o trofeos para recordarme lo bien que he jugado

Pero no solo eso, porque de un modo objetivo, el "Hotel[2]" es solamente un juego de azar con una mínima parte de estrategia en el que se va moviendo una cantidad (el dinero) de un lado a otro hasta que uno se queda con todo. Los juegos también están diseñados artísticamente para ser divertidos, y si pensamos en los grandes juegos que han triunfado entre los usuarios, todos son muy potentes en uno de los dos ámbitos: Elementos o diseño (o los dos).

2.2 Porqué aplicar la Gamificación

Estudiar y aplicar la gamificación es una práctica empresarial emergente, un "nuevo concepto de negocio" [12]. Los juegos son una herramienta muy poderosa, y es muy interesante saber qué los hace tan atrayentes y cuál es la razón de que enganchen tanto a la gente, y para ello debemos aprender de su psicología, diseño, estrategia y tecnología, ya que aplicar de forma efectiva la gamificación no es algo trivial.

También es interesante saber que la gamificación no es hacer de todo un juego, como un mundo en 3 dimensiones, ni un uso del juego para marketing o empresas si no supone un cambio en la experiencia de usuario, ni tampoco poner puntuaciones y rankings a todo, sino que se basa en aprender sobre la motivación humana y el comportamiento de las personas a través de los juegos; a través de la diversión.

[2] http://www.hasbro.com/es_MX/shop/details.cfm?R=25292110-5056-9047-F552-8974FBEAC7A2:es_MX

2.3 Juegos

Lo primero que deberíamos definir es **qué es un juego,** algo muy difícil, ya que es de esos conceptos que todos sabemos lo que significan pero que a la hora de intentar dar una definición estricta y coherente nos encontramos con que es muchas veces demasiado subjetivo y general para explicarlo. Según la Real Academia de la lengua española, jugar es el "ejercicio recreativo sometido a reglas, y en el cual se gana o se pierde." [13] Es decir, todo juego tendrá una serie de normas que los jugadores tendrán que respetar para desarrollar la realidad del juego, que está delimitada por estas reglas.

Sin embargo, para acercarnos más a la completa definición del juego podemos decir que un juego se compone de lo siguiente:

- **Objetivo:** La finalidad que tiene el juego, que puede ser diferente para jugadores de un mismo juego.
- **Reglas:** Limitaciones de la libertad dentro del propio juego.
- **Actitud de juego**: Significa que los participantes quieren jugar y quieren seguir las reglas del juego.
- Voluntad de **superar obstáculos** o retos que son en realidad innecesarios.

Todos los juegos que nos imaginemos, desde el escondite hasta el *World Of Warcraft*[3] (Ilustración 1Ilustración 1: World of Warcraft http://imagenes.es.sftcdn.net/es/scrn/47000/47837/world-of-warcraft-8.jpg) tienen estos elementos, que hacen de él una especie de burbuja al margen del mundo real donde las reglas establecidas son lo que importa. El juego por tanto es un sistema estructurado donde hay problemas, resoluciones, resultados y decisiones significativas que son interpretables, todo ello con una actitud de juego, empezando desde una salida y llegando a un final, y con una libertad y su estructura balanceada para que el jugador pueda tomar decisiones dentro de unos límites que marca el juego.

[3] http://eu.battle.net/wow/es/

Atendiendo por ejemplo al escondite, uno de los juegos más sencillos y populares a los que todos hemos jugado de pequeños, podemos fácilmente identificar las características anteriores. El **objetivo** es que no te encuentre el oponente, si “no la ligas”, y si “la ligas” el objetivo es encontrar a los demás participantes. Las **reglas** serían que el que la liga tiene que contar hasta cien o el número acordado, que mientras cuenta no puede mirar donde se esconden los oponentes, o que si éste ve a alguien y dice su nombre ese alguien ha sido pillado y tiene que salir de su escondite. Por supuesto una **actitud de juego**, por parte de todos los participantes, ya que los tramposos harán tarde o temprano que la gente no quiera jugar más, y la voluntad de que no me encuentren, de llegar a la pared donde el que la liga contaba y decir “por mí”.

Ilustración 1: World of Warcraft
http://imagenes.es.sftcdn.net/es/scrn/47000/47837/world-of-warcraft-8.jpg

2.4 Un poco de historia

En la historia reciente lo más destacable en cuanto a ocio son los videojuegos. En la década de los años 70 fue cuando se empezaron a desarrollar los videojuegos como tal, para las famosas máquinas recreativas. Juegos como el

Pong[4], mundialmente conocido y considerado la génesis de los videojuegos, calaron en la sociedad rápidamente y supusieron el inicio de la industria del videojuego, llenando de público los salones recreativos con juegos como *Space Invaders*[5] o *Asteroids*[6] (Ilustración 2).

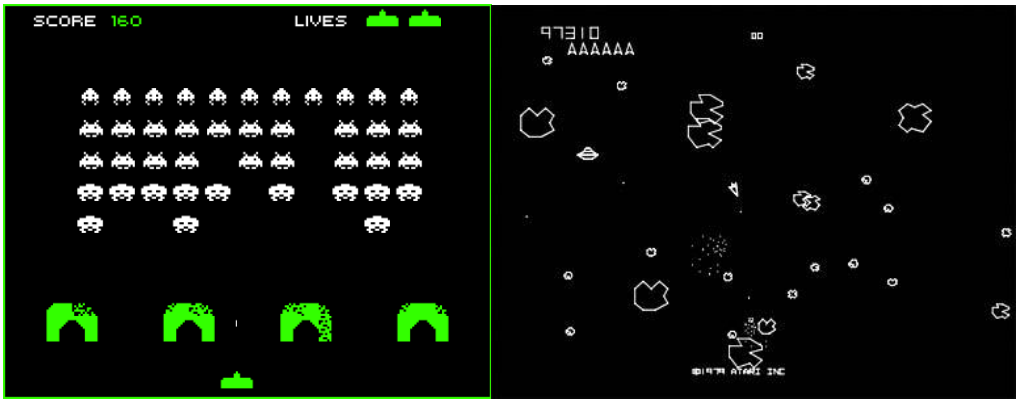

Ilustración 2: Primeros videojuegos. Space Invaders (Izquierda) y Asteroids (Derecha)
Imagenes tomadas de:
http://nodebox.net/node/documentation/concepts/xsubnetworks-space-invaders.png.pagespeed.ic.Jk6T8p5UvN.png
http://1.bp.blogspot.com/__X1iRd2Ov00/TGbEiyyBqOI/AAAAAAAADt0/VG9mwRZRtMk/s400/asteroids-atari.jpg

En la década de los 80, los videojuegos evolucionaron considerablemente debido a la gran popularidad de los salones recreativos, que alentaban a las compañías a innovar y mejorar sus videojuegos para satisfacer las demandas de un público cada vez más y más grande. De esta época los juegos más importantes son los llamados "*8 bit*", con clásicos como el *Tetris*[7] (Ilustración 3), y otros sistemas como *NES*[8] o *Master System*[9].

Ilustración 3 Primera versión del Tetris 1984
Imagen obtenida de:
http://www.unairequejo.com/blog/wp-content/uploads/2012/12/2_tetris-e1354393982425.png

[4] http://www.ponggame.org/

[5] http://www.space-invaders.com/som.html
[6] http://www.freeasteroids.org/welcome/
[7] http://tetris.com/
[8] http://www.retrones.net/
[9] http://segaretro.org/Sega_Master_System

Gracias a la competencia que generó el salto técnico que supuso el desarrollo de juegos 16 bit, los videojuegos empezaron a mejorar notablemente hasta la aparición del 3D y los videojuegos portátiles. Desde entonces, las grandes compañías como Sony[10], Nintendo[11] y posteriormente (2001) Microsoft[12], son las que dominan el mercado de las videoconsolas, y en la actualidad los videojuegos han evolucionado de forma radical y el número de usuarios ha crecido exponencialmente hasta alcanzar cifras increíbles hasta ahora.

Un factor muy importante en el aumento de los usuarios ha sido la aparición de los *Smartphones*, y con ellos la posibilidad de tener videojuegos portátiles en el teléfono móvil. Paradójicamente, los videojuegos que ofrecen hoy en día las plataformas Android o iOS vuelven a tener muchas de las características propias de los juegos *arcade* de los años 80 y 90, con lo que se puede decir que la accesibilidad y la sencillez de esos juegos *arcade* que llevan enganchando al público desde entonces, se ha impuesto en este campo a las últimas técnicas de desarrollo 3D propias de las grandes compañías de videojuegos, con lo que se ha abierto un mercado muy importante para empresas más pequeñas, que está teniendo un éxito indiscutible.

Ilustración 4. CityVille.
http://imagenes.es.sftcdn.net/es/scrn/315000/315740/cityville-63.jpg

[10] http://www.sony.com/
[11] http://www.nintendo.com/
[12] http://www.microsoft.com/

Estos juegos son, por ejemplo, *City Ville* (Ilustración 4), un juego que consiste en desarrollar una ciudad virtual y hacerla crecer y progresar desde pueblo a ciudad, que alcanzó los 100 millones de usuarios en diciembre de 2010 [2], una cantidad de usuarios tremenda y que resulta muy interesante si la comparamos con los 140 millones de personas que compraron alguna de las entregas del videojuego online más importante actualmente, "*Call of Duty*" [3].

Ahora mismo, la industria de los videojuegos factura aproximadamente el doble que Hollywood, y en china nada menos que 400.000 personas viven de jugar a *World of Warcraft*. [4] Estamos ante una de las industrias más grandes que existen ahora mismo en el mundo.

2.5 Tocando emociones

Viendo las cifras anteriores, cabe preguntarse: ¿Por qué los juegos tienen tanto éxito? ¿Por qué enganchan tanto? O de otra forma, ¿Por qué algunos juegos muy sencillos tienen muchísimo éxito y otros más complejos casi ninguno? Podríamos fijarnos para responder a estas preguntas en uno de los juegos más vendidos de todos los tiempos, el *Super Mario Bross*[13], un juego que aún hoy en día no es superado en ventas por ninguno de los juegos más sofisticados en los que se han invertido cantidades astronómicas de dinero.

Super Mario Bross es un juego de plataformas sencillo, con una historia muy poco complicada, que no tiene muchas opciones para el jugador y cuyos gráficos son 8 bit. Sin embargo, es uno de los juegos que más ha triunfado en la historia. ¿Qué ofrece *Super Mario Bross* que lo hace tan exitoso? Pues para cualquiera que haya jugado la respuesta sería: Porque es muy divertido. Es divertido sortear los obstáculos, pasar de pantalla, conseguir monedas y habilidades con las setas que se puede comer *Mario*. No es un juego en el que haya que invertir mucho tiempo en aprender a jugar, porque es sencillo, y

[13] http://mario.nintendo.com/

cualquier persona, desde los más pequeños a los más mayores, lo encuentra entretenido y adecuado.

La diversión es algo abstracto que no sabemos muy bien definir, pero que nos gusta, nos hace sentir bien y lo buscamos a veces hasta sin darnos cuenta. Cualquier cosa cuanto más divertida sea es mejor; más atrayente. Lo que nos ofrecen los juegos, como ganar, superar retos, explorar, relajarse, coleccionar, personalizar, imaginar, sorprenderse, triunfar frente a los que pierden, asumir roles, compartir (a veces), trabajar en equipo, obtener reconocimiento… es exactamente eso, diversión, y eso es lo valioso de los juegos, lo que debemos usar.

2.6 Elementos de los juegos

Todo juego tiene unos elementos que pueden estudiarse. Si nos fijamos en el clásico Tres en Raya, veremos que estos elementos son: El tablero, las fichas (*X* ó *O*) y dos jugadores. Pero la experiencia que proporciona este juego es simplemente diversión, no tiene un progreso, siempre es lo mismo, y por tanto para nosotros es un juego malo, simple. No engancha al jugador.

Los elementos dentro de los juegos son múltiples y variados, y dependiendo de los que usemos y a cuales demos más importancia conseguiremos un tipo de juego u otro. Así, podemos considerarlos como una pirámide, la “Pirámide de los elementos de Gamificación” (Ilustración 5), que se compone de tres grupos diferenciados por su naturaleza. Los elementos por tanto se dividen en dinámicas, mecánicas y componentes [4].

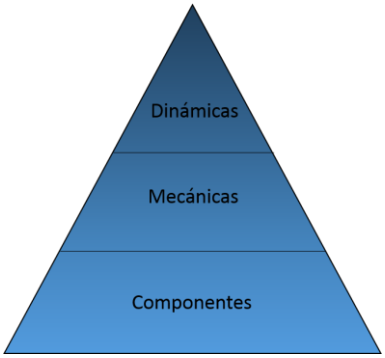

Ilustración 5. Pirámide de los elementos de la Gamificación

2.6.1 Dinámicas

Las dinámicas son los elementos de más alto nivel, tienen que ver con la estructura del juego, que creará la burbuja de realidad dentro del juego que tendrá sentido para el jugador. Son las reglas del juego, aunque no solo eso. Así tendremos:

Restricciones, que dibujan los límites de la libertad del jugador. Muy importantes para poder crear la dinámica de juego. Hay que saber establecer bien las restricciones ya que demasiada libertad puede abrumar al jugador, y que éste no sepa bien qué objetivos debe seguir, y una libertad demasiado limitada puede derivar en una actividad aburrida y carente de interés y motivación.

Emociones que suscita el juego, ya sea diversión, felicidad, sorpresa o incluso enfado. Las emociones es lo que hacen que un juego triunfe o fracase.

Narrativa, que puede ser contada por el juego o venir implícita en él. Une las piezas del juego y hace que éste fluya, que sea consistente, por lo que es muy importante que el juego tenga coherencia, que haya una lógica en los objetivos.

Progresión, Muy importante para que el jugador tenga la sensación de avance, de mejora. Una buena progresión se puede conseguir haciendo uso de los componentes que se explicarán más adelante.

2.6.2 Mecánicas

Las mecánicas son como los verbos del juego, son las que hacen que la acción se mueva hacia delante, que el juego siga avanzando. Son muchas más que las dinámicas y entre ellas tenemos:

Retos o challenges: Objetivos a alcanzar dentro del juego.

Azar: Un resultado aleatorio que el jugador en principio no puede controlar. En ocasiones puede resultar útil y dar un toque diferente al juego para que no sólo dependa de la habilidad del jugador.

Competición y cooperación con otros jugadores. Trabajo en equipo

Feedback: Usado para que seguir jugando sea algo que se estimula desde el propio juego. Un buen uso del *feedback* dentro de una actividad lúdica deriva en una respuesta satisfactoria del jugador, que lo animará a continuar.

Adquisición de recursos, como oro, madera… que nos permiten mejorar dentro del juego. Esta mecánica se usa mucho en los juegos que actualmente más de moda están entre las aplicaciones móviles. Por ejemplo, *Clash of Clan*s, el juego más descargado actualmente en el *AppStore* y que más beneficios ha obtenido [5], se basa en la mejora a través de recursos.

Premios: El sistema para obsequiar al jugador por conseguir determinadas cosas en el juego.

Transacciones de recursos: Para mejorar gastándolos en el juego.

Turnos que proporcionan un orden lógico del juego.

Todas las mecánicas se pueden resumir en un solo verbo: **MOVER** al jugador para que siga jugando, para que avance y para que descubra todas las emociones que el juego puede darle. Por supuesto, no todas las mecánicas han de ser usadas en un juego ni el uso de la misma combinación nos dará siempre el mismo resultado.

2.6.3 Componentes

Por último, los componentes es la parte más extensa de los juegos, donde más elementos hay, y vienen a representar los sustantivos del juego, lo tangible, lo que el jugador verá. Entre otros están:

Logros: Una recompensa o premio por haber conseguido algo concreto.

Avatar: La representación del jugador dentro del juego

Badges: Una representación visual de los logros obtenidos (Ilustración 6). Ver

apartado 2.8

Jefes Finales: Retos difíciles que aparecen al final de los niveles

Coleccionables: Elementos que forman parte de un todo y que se consiguen poco a poco.

Contenido desbloqueable: Que se desbloquea tras una serie de logros o acciones.

Ilustración 6: Varios ejemplos de Badges
Imagen obtenida de: http://2.bp.blogspot.com/-RVKdiziQhBs/UTSddnI3tMI/AAAAAAAAArc/_eNWh1qcf6Y/s1600/Badges.png

Regalos: Pequeños obsequios que podemos dar a otros jugadores

Leaderboards: Paneles informativos que ofrecen una visualización de mi progreso respecto al de los demás, dando una sensación de competitividad.

2.7 La importancia de los elementos

Si observamos los apartados 2.6.1 a 2.6.3, podemos llegar a la conclusión de que tenemos una gran cantidad de opciones con estos elementos, que se pueden combinar de muchas maneras. Los elementos de los niveles de abajo sirven para completar los elementos de los niveles de arriba. A través de niveles y leaderboards se genera competición, y con un sistema de premios y

feedback se genera progresión en el juego.

Si cogemos un juego cualquiera, como el Palabraz[14] (Ilustración 7), un juego que se basa en encontrar el mayor número de palabras combinando letras que aparecen en un tablero en un tiempo determinado, veremos que sus componentes son: Puntos por conseguir palabras, niveles que se van alcanzando cuanto más jugamos, Leaderboards permanentes y en cada partida, que estimulan la competición, un sencillo grafo social en el que podemos seguir a la gente que queramos, premios o badges que se consiguen por alcanzar metas dentro del juego… etc. Las mecánicas del juego son básicamente la competición con los demás usuarios por conseguir el mayor número de puntos en cada partida y subir puestos en el ránking del país donde juguemos. Por último, la dinámica serían las reglas que debemos seguir, y que están muy bien definidas, como el tiempo que tenemos para encontrar las palabras, la manera de formarlas, solamente juntando casillas adyacentes y sin levantar el dedo de la pantalla… etc.

Ilustración 7: ***Palabraz***
http://static1.appsda.com/ios-screenshot/palabraz_4f35625025737_full.jpg

No todos los elementos tienen la misma importancia, y dentro de la

[14] https://play.google.com/store/apps/details?id=tr.com.fugo.kelimeavi2.es

gamificación hay unos elementos que son mucho más importantes que otros, ya que con ellos se pueden lograr casi todos los objetivos de la gamificación, aunque no se tienen porqué usar sólo esos.

Estos elementos forman la llamada “Triada PBL”.

2.8 La triada PBL

La llamada “Triada PBL” es el conjunto de los tres elementos más importantes de la gamificación: puntos, *badges* y *leaderboards*. Únicamente combinando estos 3 elementos se puede cubrir la mayor parte de la estructura gamificada.

2.8.1 Puntos

Los puntos determinan cómo de bien me está yendo en el juego, cuánto he jugado y además es un dato fácilmente comparable con el de los demás usuarios. Con ellos se pueden generar niveles, se pueden usar para el feedback, podemos conectarlos a los premios, para que estos se desbloqueen cuando lleguemos a un número determinado de puntos, pueden ser la visualización del progreso del juego o pueden ser un dato que utilice el diseñador del juego para gestionar los niveles de acceso de los usuarios.

2.8.2 Badges

Los *badges* son las indicaciones de objetivos alcanzados. Nos muestran los logros que hemos conseguido en el juego con algo tangible que normalmente representa la estética del juego (ver Ilustración 6). Con los *badges* podemos saber qué es importante conseguir en el juego, qué es premiable, y dan credenciales al usuario para que se sepa qué cosas ha logrado. Éstos se pueden combinar con otros elementos de los nombrados anteriormente como

los coleccionables y crear diferentes temáticas de *badges* diferenciadas por el tipo de objetivo, dependiendo de lo complicado que queramos que sea el sistema para otorgarlos. Lo importante de estos elementos es su capacidad para hacer más atractivo el juego, de forma estética, pero también proporcionando una sensación de continuidad y dando al jugador una motivación para seguir jugando: conseguir todos los premios. Un ejemplo muy claro son los "logros" de la videoconsola *XBOX 360*, donde todos sus juegos tienen unos *badges* desbloqueables únicos para cada juego que se desbloquean cuando conseguimos ciertos hitos o habilidades dentro del juego y que, asociados a unos puntos, forman parte de nuestro perfil y nos colocan dentro de un ranking de experiencia de jugador que se puede comparar con otros usuarios a través de XBOX Live.

2.8.3 Leaderboards

Los *leaderboards*, o tableros de clasificación (ver Ilustración 8), son los que hacen la función de *feedback* dentro del juego. Dan la sensación de competitividad y progreso. Hay que tener cuidado porque un mal uso de los *leaderboards* puede terminar quitándole atractivo al juego ya que si el rival a batir está demasiado avanzado como para tener siquiera la oportunidad de vencerle, no merecerá la pena intentarlo, y por eso una buena práctica es usar *leaderboards* personalizados, relativos, que nos dejen compararnos con la gente que está en un rango de niveles, o por zona geográfica, o por círculo de amigos.

Ilustración 8: *Leaderboard* de "League of Masters"
http://igmarauders.isotx.com/library/images/4/48/Leaderboard.png

2.9 La Gamificación no lo es todo

Se puede pensar que con los elementos anteriores se puede crear cualquier sistema de gamificación y tener éxito, y esto no es así. Los elementos no son lo único que hace la gamificación, no son el juego, ya que no todos los premios son divertidos de conseguir, y no toda la diversión se basa en premiar. Un buen juego y, por tanto, una buena aplicación de la gamificación es la que contenga, a través de los elementos anteriores, una experiencia para el usuario que le permita divertirse a través de sus propias elecciones dentro del sistema, y que pueda hacerlo en comunidad y con diferentes tipos de usuarios.

Capítulo 3: Requisitos

3.1 Introducción

Como hemos visto, la gamificación es una técnica que se puede usar para estimular la motivación de los usuarios y conseguir unos objetivos. Desde una aplicación muy sencilla hasta un complejo sistema de niveles, puntos y badges, cada vez son más los entornos (que nada tienen que ver con los juegos) que aplican esta filosofía "*game like*", ya que sus resultados son significativamente positivos en cuanto a uso y "enganche" de los usuarios.

Si lo vemos desde el punto de vista de la programación de aplicaciones, sería muy útil disponer de una arquitectura de gamificación que pusiera a nuestro alcance las herramientas suficientes para que se pudiera aplicar una estrategia de este tipo en nuestra aplicación. Separar, desde el punto de vista técnico, las funcionalidades que en sí ofrece nuestro proyecto de las que forman parte de la gamificación propiamente dicha. Así, se podría disponer de un *framework* que ayude a los desarrolladores a aplicarla sin tener que cambiar la estructura básica de su aplicación, disponiendo así de los elementos que hemos visto anteriormente para poder hacer uso de ellos de forma paralela.

Así, podemos estudiar, desde ese punto de vista de programador de aplicaciones, qué requisitos debe tener una arquitectura de gamificación para poder cubrir las posibles necesidades. Para ello, vamos a atender a dos escenarios diferentes que pretenden incluir esta arquitectura y las soluciones, diferentes en cada uno de ellos, que serían satisfactorias.

3.2 Escenario 1: WIKI Corporativa

En este primer escenario se presentará la herramienta WIKI, un software que se utiliza en las empresas y que parece una buena aplicación para adaptar un sistema de gamificación.

3.2.1 Introducción y contexto

Dentro de una gran empresa de tecnología, en la división de producción de máquinas especiales, se ha introducido recientemente la herramienta de la WIKI para compartir conocimiento entre los internos y que las lecciones aprendidas en el desarrollo del trabajo perduren en el tiempo y que el conocimiento dependa menos de la plantilla actual de la empresa, cosa que siempre ha supuesto un problema para las nuevas incorporaciones, y por tanto para la empresa, que hasta la existencia de esta aplicación compartían este conocimiento mediante complicados sistemas de archivos y carpetas alojados en los servidores locales. Esta complicación llevaba en muchas ocasiones a los empleados a imprimir y guardar la información de forma física, dando problemas de almacenamiento, aumentando los gastos de impresión y los riesgos de seguridad de la documentación. Para ello, la empresa decidió invertir en una aplicación web que se pudiera utilizar para compartir la información más relevante para los internos de una forma mucho más sencilla y accesible.

La WIKI es una plataforma web interna, no accesible desde fuera de la empresa, que sólo puede utilizar la plantilla actual. Cada uno de los usuarios de la WIKI tiene diferentes grados de libertad para publicar artículos y editarlos, dependiendo de su departamento y cargo dentro de él, con un sistema para comentar y puntuar las publicaciones de los demás internos. Además, el sistema cuenta con un apartado de estadísticas donde se pueden visualizar las visitas realizadas en un periodo de tiempo determinado y la actividad en general que ha tenido la herramienta.

Se pretende que en un futuro próximo la información relevante para los empleados (como los manuales de uso de programas informáticos, o instrucciones para la calibración de instrumentos de medida) se aloje en su totalidad en esta herramienta, por lo que la empresa ha dedicado recursos para la formación de los empleados, pero no se han conseguido de momento los resultados que se pretenden, ya que aunque la WIKI lleva implantada varios meses, su uso aún es bastante escaso. Los problemas que se están observando es que aunque hay bastantes usuarios que ponen esfuerzo en publicar artículos interesantes para su departamento, las visitas son aún muy escasas según las estadísticas, y a pesar de las formaciones todavía hay quien no sabe usar la aplicación del todo, así que siguen trabajando como hasta ahora.

3.2.2 Características requeridas

Se ha pensado en implantar a la herramienta WIKI un sistema de gamificación para motivar su uso y conseguir mejorar esas estadísticas. Los requisitos de este sistema contienen los elementos que se presentan a continuación.

Usuario editable: Para personalizar cada usuario, con un avatar a elegir entre unas imágenes corporativas predeterminadas, y que contenga la información básica del empleado, como nombre y apellidos, departamento, dirección de contacto y teléfono… etc.

Badges: Se pretende premiar al usuario dándole medallas virtuales que se puedan coleccionar con un estilo corporativo por completar tareas como las siguientes:

- Terminar de completar toda la información requerida en el perfil.

- Ser el usuario que más publicaciones ha hecho en un periodo de tiempo determinado. Habría, por tanto, premios para publicista de la semana, del mes o del año.

- Ser uno de los que más comentarios reciben en sus publicaciones en un periodo de tiempo determinado, también semana, mes o año.

- Cada vez que se llegue a un número de visitas concreto, sin repetir artículos, y el tiempo de permanencia en el artículo supere el minuto. Deberían ser al menos para las 10, 20 y 50 publicaciones visitadas.

- Haber recibido un número determinado de comentarios en tus artículos. 5, 10, 20.

- Haber comentado un número determinado de artículos. 5, 10, 20. Se presupone que las publicaciones serán relevantes, ya que la herramienta es de entorno empresarial y podrá ser siempre visualizada por los superiores. Además, las publicaciones podrán ser puntuadas negativamente.

- Hacer actualizaciones, tanto de los artículos propios como los de los demás internos. 5, 10, 20.

- Añadir palabras clave para facilitar la búsqueda a un número determinado de artículos. 10, 20, 50.

***Leaderboards*:** Para motivar la competitividad entre los usuarios de la aplicación, se pretende introducir también tablones donde se pueda ver el progreso relativo del usuario, comparándose así con los demás integrantes del departamento, o con todos los participantes. Se deberían tener al menos los siguientes *leaderboards*:

- Número de publicaciones global

- Número de publicaciones por departamento.

- Número de visitas que ha recibido cada usuario en sus artículos, global.

- Número de visitas que ha recibido cada usuario en sus artículos, por departamento.

- Número y porcentaje de *Badges* conseguidos.

- Valoración Global. Mediante un cálculo ponderado de los logros anteriores.

***Teams*:** Para poder distribuir a la plantilla por grupos que tengan diferentes tipo de acceso a los apartados y publicaciones de la aplicación, se deben definir *Teams* por cada departamento. Así, el departamento de mecánica tendrá acceso de lectura y escritura en el apartado "EDM", que es el de procesos mecánicos, y sólo de lectura en los de "EES", que es el apartado de procesos eléctricos.

Sería conveniente que además la arquitectura de gamificación que se implante cuente con **notificaciones** cuando se consigan alguno de los logros que se han mencionado antes, para que se motive visualmente al usuario. Por ejemplo, un panel superior que indique el porcentaje de finalización del perfil. (nos recuerda siempre que no hemos terminado de incluir toda la información posible e invita a completarlo, sugiriendo el siguiente paso), un tablón lateral con los *badges* conseguidos en miniatura, coloreados los que están conseguidos, con una interrogación los que aún nos faltan por conseguir y pistas para poder conseguirlos.

El primer contacto con el sistema de gamificación para el usuario final debe ser sencillo, de fácil incorporación, y que al principio le sea muy fácil conseguir los *badges*. Además, todos los gráficos que se incluyan deben tener un estilo corporativo, para seguir la línea gráfica de la empresa.

3.3 Escenario 2: Aplicación Móvil

Actualmente existe una aplicación móvil para varias plataformas llamada *Runtastic*[15]. Esta App permite al usuario registrar información acerca de las actividades deportivas que realiza. Por ejemplo, si el usuario usa la aplicación

[15] https://www.runtastic.com. Desde la página principal se puede descargar la aplicación para varias plataformas móviles.

para hacer una maratón, puede registrar el tiempo que ha tardado, el recorrido impreso en un mapa, la sensación que ha tenido, el ritmo medio, la altitud y demás información configurable, y toda esa información se puede compartir mediante redes sociales y la propia red social *Runtastic* que provee la aplicación. Tiene además con la versión Premium, por la que hay que pagar, mensajes motivacionales que ayudan al deportista. Le informan de lo que lleva recorrido, lo que le falta… etc. (ver Ilustración 9)

Esta aplicación es bastante popular dentro del mundo del deporte, sobretodo amateur, pero el problema que presenta es que su uso no parece que esté perdurando en el tiempo para los usuarios, ya que no se puede mejorar dentro de la aplicación, y se convierte simplemente en un registro de actividades que se puede consultar. Muchos de los usuarios terminan olvidando activar la aplicación antes de hacer la actividad deportiva o simplemente no ven la utilidad de registrarlo.

Se ha pensado en una estrategia de gamificación para que el usuario se enganche más a la aplicación y haya más *feedback* entre los usuarios. Así habrá más descargas de la aplicación y será más atractiva la versión Premium, por la que los desarrolladores obtendrán más beneficio.

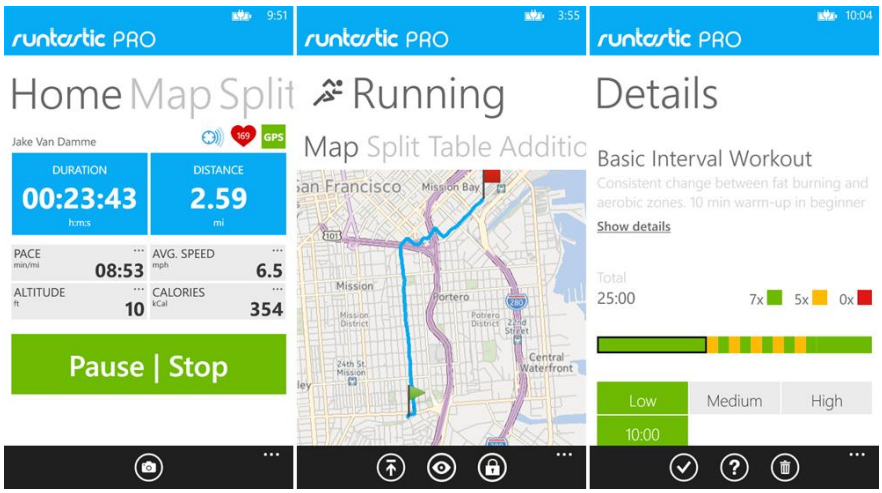

Ilustración 9 Interfaz de la aplicación *Runtastic*
Imagen obtenida de: http://www.aptoide.info/wp-content/uploads/2014/04/runtastic-pro-screenshot-1.png

La gente que se supone que utiliza esta aplicación es gente que le gusta el deporte y disfruta mejorando sus marcas, está en contacto con la naturaleza y le gusta disfrutar de su tiempo libre, así que se quiere que en la temática y en el diseño de los elementos de la arquitectura de gamificación se reflejen estos aspectos.

3.3.2 Características requeridas

Necesita las siguientes características:

- **Trofeos**: Desbloqueables por terminar diferentes actividades. Por ejemplo: trofeo ciclista, al superar un número de carreras con la bicicleta, pudiendo obtener el trofeo de bronce, plata y oro.

- **Puntos por actividades**, que puedan dar una idea de qué deportes hemos practicado más, en cuáles somos más constantes y tenemos más experiencia para poder comparar con los diferentes usuarios de la aplicación. Casi cualquier uso de la aplicación debe subir puntos, ya que todo esfuerzo debe ser recompensado, y esas recompensas tienen que ser acordes con la dificultad de la actividad realizada. Los puntos se dividirán por actividad y además habrá una puntuación global que vendría a indicar el grado de implicación con la aplicación que tiene el usuario, clasificando a los usuarios a partir de esa puntuación en “Principiante”, “Amateur”, “Experimentado”, “Alto rendimiento” y “Élite”. Cada nivel es más difícil de conseguir que el anterior y más fácil que el siguiente.

- **Usuarios**: Personalizables con avatar o foto, información, intereses deportivos, etc. El usuario es el escaparate exterior que tendrá el que utilice la aplicación y por eso es importante que se pueda personalizar lo máximo posible, dentro por supuesto de unos límites intrínsecos de la propia

arquitectura para que todos los perfiles tengan unos puntos en común que los hagan distinguibles de los de otras aplicaciones. Las ediciones de los perfiles vendrán sujetas a una serie de permisos que se irán desbloqueando a medida que se vaya mejorando dentro de la aplicación. Así, un usuario "Alto rendimiento" tendrá más opciones de edición que otro usuario "Amateur", con lo que motivaremos la mejora en la aplicación para conseguir estos privilegios de edición.

- **Grafo social**, que nos permita estar siempre conectado con la gente que nos interesa. Estas conexiones se harán a veces de forma directa, cuando un usuario agregue a un conocido a su lista de amigos, o de forma indirecta, cuando varios usuarios tengan una misma característica en su perfil como el centro deportivo al que pertenecen, zona geográfica…

Para que el grafo social nos permita además hacer contactos y amistades, sería muy interesante que hubiera una opción que nos permita hacer una búsqueda según afinidades, es decir, los usuarios cuyo perfil más se asemeja al nuestro en cuanto a deportes practicados, tiempos de entrenamiento… etc.

- ***Leaderboards***: A partir de los trofeos conseguidos, los puntos por actividades, y demás características y objetivos superados por el usuario se definirán tablones en los que se pueda observar a modo comparativo con los demás deportistas ese progreso individual, teniendo varios leaderboars, tanto generales como por amistades o por deporte.

3.4 Requisitos de una arquitectura de gamificación

Tras haber hecho una amplia descripción de los requisitos de la posible arquitectura de gamificación en los anteriores escenarios, podemos hacernos una idea de las características que debería tener un sistema genérico de gamificación que pudiera dar servicio a diferentes aplicaciones. Si bien no todos los supuestos en los que se necesitará esta arquitectura serán iguales en

cuanto a tecnología utilizada o filosofía de gamificación, la solución técnica que propongamos tiene que ser capaz de cubrir las diferentes necesidades que se presenten, partiendo de unos puntos en común que serán útiles en todas las aplicaciones, y llegando a funcionalidades aplicables o no en función de los objetivos y la naturaleza de las plataformas donde se utilice.

3.4.1 Diagrama general

En primer lugar, debemos tener claro el esquema general (Ilustración 10) presentará la solución, que debido a su versatilidad tendrá que ser diferenciable de la aplicación, ampliando la funcionalidad de la misma tras su implementación.

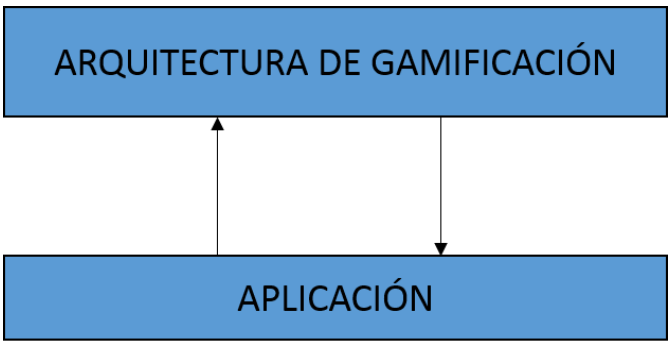

Ilustración 10 Diagrama básico

De esta forma, el desarrollador de aplicaciones no tendrá que tener en cuenta el sistema de gamificación desde el comienzo del desarrollo software, sino que podrá aplicar la arquitectura a posteriori, como un *framework* que añade a las funcionalidades de su aplicación para actualizarla. Por tanto, la gamificación se gestionará al margen del programa, proporcionando al desarrollador total libertad en la implementación y quitándole carga de trabajo ya que no tendrá que preocuparse por esta nueva funcionalidad, sino que simplemente hará uso y se beneficiará de las características que ofrece, ajeno totalmente a su funcionamiento interno si lo desea, ya que este será el objetivo de nuestra arquitectura: proporcionar una solución técnica de implementación de gamificación en las aplicaciones a los desarrolladores de aplicaciones.

3.4.2 Partes de la arquitectura

En líneas generales, la arquitectura de gamificación como hemos visto debe ser capaz de registrar un conjunto de usuarios con sus datos asociados y una serie de elementos de gamificación que hagan posible su funcionamiento. Debe también proporcionar un motor para poder modificar esos datos desde la aplicación a través de una serie de órdenes o funciones y devolver resultados para su lectura o visualización en el programa donde se aplique. (Ver Ilustración 11).

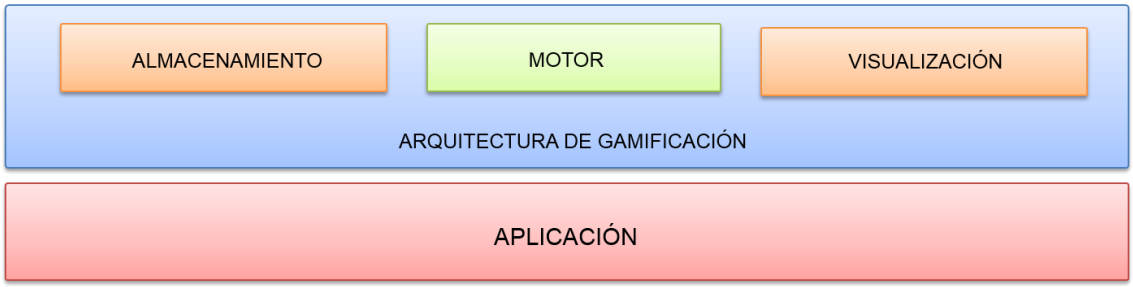

Ilustración 11: Partes de la Arquitectura

3.4.2.1 Almacenamiento

El salvado de datos y recursos hay que enfocarlo desde un principio. Hay muchas opciones para almacenar contenido de aplicaciones y la que se elija dependerá de qué tipo de servicio se quiere ofrecer a los programadores, qué magnitud de datos se va a manejar teóricamente y, en menor medida, a qué tipo de aplicaciones queremos enfocar nuestra arquitectura.

Las soluciones que podemos utilizar podrían ser un servidor de uso, como pudiera ser útil en una aplicación de entorno empresarial, o un servidor web, si la aplicación va destinada a un público menos limitado. La base de datos puede ser lo complicada que queramos, aunque con una base de datos MySQL sencilla podría valer.

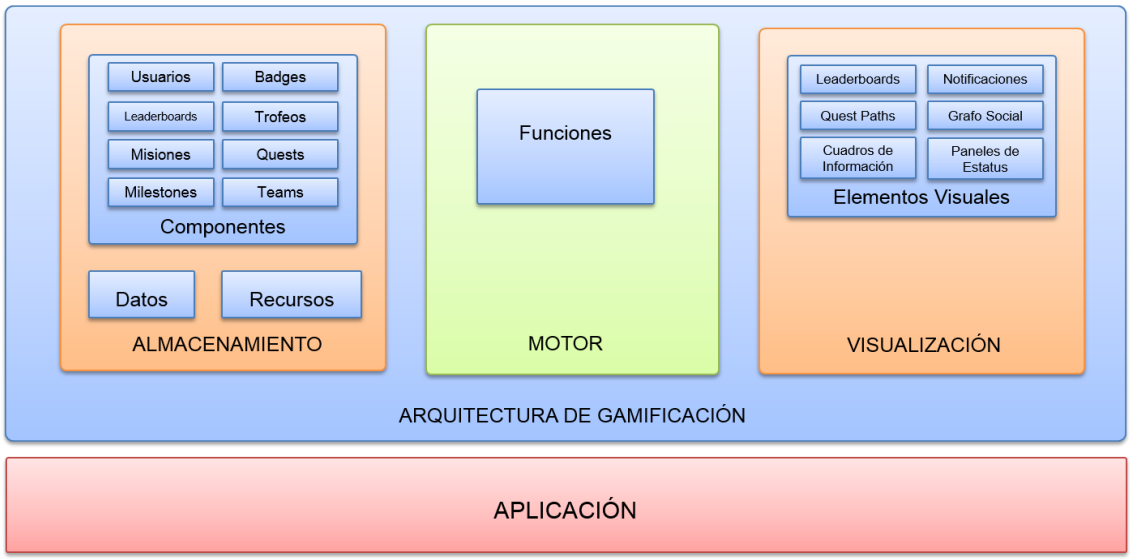

Ilustración 12 Elementos de la arquitectura

Los usuarios y sus datos asociados son elementos dinámicos, es decir, estarán sujetos a constantes modificaciones por parte de los usuarios finales, mediante el uso del programa. Por tanto, la arquitectura debe permitir al programador de aplicaciones insertar las funciones correspondientes para que estos cambios se lleven a cabo. Por otro lado, los elementos de gamificación, llamados componentes, serán estáticos, es decir, se usarán como recursos a los que se acceda únicamente para su lectura. Estos son, por ejemplo, los *badges*, los cuales están creados desde el principio y no son modificables, pero sí asignables a los diferentes usuarios.

Aunque no están definidos cuáles son todos los componentes (ver apartado 2.6.3) necesarios para que una arquitectura de gamificación pueda funcionar, debido a que eso depende en gran medida de la naturaleza de la aplicación donde se vaya a implantar, podemos, a partir de los conocimientos obtenidos en el estudio del arte de la gamificación, proponer los siguientes: *Usuarios,*

Badges, Leaderboards, Trofeos, Misiones, Quests, Milestones y Teams. (Ilustración 12).

3.4.2.2 Visualización

Para que el sistema de gamificación sea algo apreciable por el usuario final, es necesario disponer de una serie de elementos visuales que den la información que corresponda en la aplicación. Esta información vendrá con la presentación que el desarrollador considere adecuada y que se podrá obtener a partir de la arquitectura.

Para gestionar tanto la parte del almacenamiento como la de la visualización, es necesario disponer de una lógica dentro de la arquitectura que los conecte con la aplicación, el motor.

3.4.2.3 Motor

La lógica de la arquitectura debe estar presente tanto en la parte del almacenamiento como en la parte de la aplicación en sí, ya que aunque la mayor parte de las acciones que tengamos que realizar para modificar datos en el almacenamiento vendrán dadas desde el cliente, como hemos visto cliente y servidor no tienen por qué estar programados en un mismo lenguaje y habrá determinadas partes de la arquitectura que se manejen desde el propio servidor. Además, la parte de servidor tiene que ser capaz de traducir las órdenes que se envían desde cliente y transformarlas en los cambios correspondientes que se están enviando. El motor debe ser capaz de dar el servicio suficiente como para que el usuario final pueda dar órdenes a la aplicación y ésta a su vez envíe esas órdenes al servidor para que eso se traduzca en una lectura o escritura de datos en el almacenamiento. Esto, traducido a código programado, significa una serie de funciones y métodos que interconecten las partes entre sí. Como mínimo, atendiendo a los requisitos de las aplicaciones de ejemplo que hemos visto en los capítulos anteriores,

deberíamos disponer en la arquitectura de las funciones que se explican a continuación.

3.4.3 Funciones

En primer lugar, las funciones que modificarían los datos almacenados serían las siguientes:

- Crear, actualizar y borrar usuario
- Puntuar positivo, puntuar negativo
- Ganar y perder *badge/s*
- Actualizar, ganar y perder *badge-point*
- Añadir y quitar elementos del grafo social
- Subir de nivel
- Subir el progreso de nivel
- Añadir y quitar de equipo
- Cambiar niveles de acceso
- Actualizar y ganar misiones
- Actualizar, ganar y perder quests
- Ganar trofeos.

Además, para cada uno de los apartados anteriores serían necesarias funciones para devolver los datos sin modificar y poder ampliar la posible gestión de los datos desde el código de la aplicación. Es decir, obtener la información de los usuarios (por ejemplo) a partir de una función en el código de la aplicación para poder luego con esos datos hacer las acciones oportunas.

Todas las funciones que hemos visto no tienen en principio por qué tener una visualización directa para el usuario final, es decir, se hacen de forma invisible

y asíncrona a la aplicación. Se mandará una de las anteriores órdenes y la arquitectura la gestionará paralelamente a la aplicación. No obstante, también habrá funciones que permitan mostrar los datos almacenados en forma de visuales para que la gamificación sea tangible para el usuario final.

El tratamiento de los visuales se puede hacer de varias formas, pero parece conveniente que todo aquello que vaya a ser visible por el usuario final sea editable, desde el punto de vista de formato, por el programador de aplicaciones, ya que éste le aplicará las características que considere oportunas para que concuerde con la estética de la aplicación. Por tanto, se ofrecerán desde la arquitectura funciones que devuelvan visuales cuyas características de tamaño, forma, colores de fondo… etc. sean modificables por el programador.

Otra opción para el desarrollador será crear sus propios visuales sin utilizar estas funciones, haciendo uso únicamente de la lectura de datos del servidor y mostrándolos de forma personalizada en su aplicación. Esta opción hará un uso menos completo de la arquitectura pero también dará más libertad a la hora de caracterizar y dar formato a los visuales.

3.5 Funcionalidad y Ejemplos

Después de haber visto todos los requisitos que debe tener una arquitectura de gamificación para dar un servicio completo a cualquier aplicación que desee aplicar esta técnica, el procedimiento que seguirían los escenarios anteriores, a modo de ejemplo, sería el siguiente:

En la WIKI corporativa debíamos premiar con un *badge* a aquellos usuarios que visitaran 10 artículos. En la arquitectura que proponíamos anteriormente, esto se traduce en un *badge-point*, ya que es un *badge* que no se conseguirá hasta alcanzar una determinada puntuación, en este caso 10. Este *badge-point* ha debido ser creado anteriormente en el servidor como un elemento caracterizado por un nombre de identificación, que podría ser “*badgeVisitas”*, la

puntuación que se debe alcanzar para conseguirlo y la imagen que lo identifica visualmente.

El usuario primero accede a la aplicación con una identificación y contraseña, que se utilizarán para hacer las modificaciones en los datos del servidor. Luego accede a un artículo. En ese momento, la aplicación hace una llamada a la función que se encarga de actualizar un *badge-point*. Dentro de los parámetros de esa función tendremos que especificar qué usuario es, qué *badge* se actualiza (en este caso "*badgeVisitas*") y cuánta puntuación se asigna a ese *badge*, en este caso un punto. Esa información se envía codificada al servidor, donde tras decodificarla, el motor accede a los datos almacenados y busca al usuario. Luego, añade una unidad al *badge* correspondiente, compara con la puntuación necesaria para ganarlo y guarda los datos en el almacenamiento. Finalmente, envía una respuesta a la aplicación confirmando los cambios realizados.

Este procedimiento se repetirá con la única diferencia de que cuando sea la décima llamada a esa función para ese *badge*, en el momento en el que el servidor compare la puntuación actual con la necesaria para ganar el *badge*, comprobará que son iguales. Ahí, almacenará además que el usuario ha conseguido desbloquear ese elemento, y la respuesta que envíe a la aplicación será diferente también, indicando así que hemos conseguido el "*badgeVisitas*". De vuelta a la aplicación, se informará al usuario de que ha conseguido desbloquearlo, y para ello se llamará de nuevo al servidor para que devuelva los datos correspondientes a ese *badge-point*. El servidor recibe esa llamada, consigue esos datos almacenados y los devuelve. En este caso los datos podrían ser el nombre del *badge*, la imagen que tiene asociada y una breve descripción de lo que ha hecho el usuario para desbloquearlo. Esta información la aplicación la recibe y la trata para presentarla de forma visual mediante una notificación que el usuario recibe.

En la aplicación "*Runtastic*" uno de los requisitos que se pedía era que se pudieran visualizar *Leaderboards* creados a partir (por ejemplo) de los trofeos conseguidos (ver apartado 3.3.2 Características requeridas).

El usuario entra en el apartado de clasificaciones de la aplicación. Previamente o en ese mismo momento (la estrategia a seguir dependerá del programador de aplicaciones), se enviará una petición al servidor a través de la función que tenga como objetivo devolver la información necesaria para rellenar el tablón de clasificación. En este caso, el programador quiere que se muestre el *Leaderboard* los usuarios a nivel mundial que más trofeos han ganado, así como la posición del usuario actual que visualizará este tablón, así que se enviará la petición personalizada al servidor, que la leerá, accederá a los datos y devolverá la información solicitada. Una vez de vuelta en la aplicación, esos datos se tratarán para su presentación visual en la aplicación.

Capítulo 4: Soluciones actuales

4.1 Introducción

Una vez conocemos los requisitos que debería tener la arquitectura de gamificación que requerimos en el anterior punto, vamos a analizar las propuestas que se ofrecen actualmente.

Para ello, nos centraremos en primer lugar en las soluciones comerciales que podemos encontrar en el mercado en estos momentos (Ilustración 13), para posteriormente centrarnos en una arquitectura de software libre y gratuito que se puede encontrar actualmente en internet: ***Userinfuser***. Aunque se expliquen a continuación más arquitecturas, muchas de ellas con un desarrollo muy amplio, nos centraremos en ésta ya que al ser su software totalmente accesible y modificable nos permitirá hacer un estudio más en profundidad y entender mejor cuál es la filosofía de esta propuesta, y posteriormente ser capaces de proponer mejoras y cambios para adaptarla a otros posibles requisitos de uso.

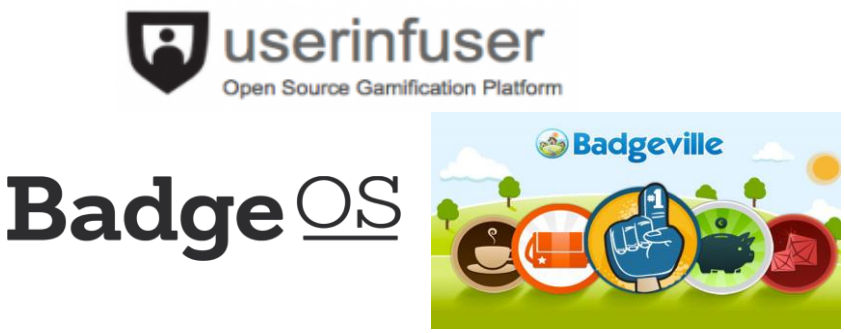

Ilustración 13: Algunos ejemplos de Arquitecturas de gamificación

4.2 Soluciones comerciales

4.2.1 *Badgeville*

Badgeville es una de las soluciones para aplicaciones que quieran implantar gamificación. En su página principal[16] podemos encontrar la información para su contratación y uso.

En un entorno comercial, donde los usuarios que vayan a utilizar este sistema gamificado sean clientes de marcas o productos determinados, *Badgeville* ofrece crear una lealtad entre los consumidores del producto. Dado que los clientes hoy en día son menos leales que nunca, aplicar técnicas que estimulen o impulsen al cliente a desarrollar una relación más comprometida con la marca que consumen favorecerá notablemente los resultados comerciales que se obtengan. *Badgeville* ofrece para ello soluciones para páginas web, aplicaciones móviles y redes sociales, impulsa la atracción de nuevos productos a través de *badges* contextualizados (Ver Ilustración 10) y otorgados por comportamientos específicos. Además, ofrece extender el comercio del producto a través de páginas web, aplicaciones móviles, redes sociales, y campañas de marketing en estas plataformas.

Para el ámbito empresarial, la máxima es que “el poco compromiso de los empleados le cuesta a la empresa empleadora dinero real” [6]. Se ofrece una mejora en la productividad y las ventas a través de las técnicas de gamificación, promoviendo también la colaboración entre los empleados de la empresa.

También se ofrece aumentar la atracción de los clientes hacia los productos de la empresa. Debido a que casi la mitad del software que se desarrolla para crear aplicaciones empresariales no se utiliza, se están perdiendo miles de millones de dinero invertido. Con *Badgeville* se ofrece reducir en un 50% esas pérdidas potenciales [6].

[16] http://www.badgeville.com

La base de *Badgeville* es la llamada "*Behavior Platform"* a través de la cual se pueden gestionar los sistemas de gamificación de todas las aplicaciones de la empresa que lo utilice, pudiendo adecuar a las necesidades y circunstancias de cada una los distintos elementos de gamificación que se ofrecen en esta plataforma. Es una herramienta que permite "medir, analizar y reconocer el comportamiento de los usuarios a través de tu ecosistema digital" [6] y así otorgar premios a través de las mecánicas de juego, dar una reputación evaluando su status, conectarlos a través de una red social y darles la oportunidad de analizar el éxito de sus acciones, a través de analíticas que puedan consultar.

Para poder usar Badgeville en cualquiera de las aplicaciones que el cliente quiera, se dispondrá de una consola con una interfaz de usuario para poder manejar los comportamientos en los que se quiera influenciar para experimentar un cambio. Por otro lado, se dispondrá tambien de herramientas de desarrollador para poder integrar Badgeville en los sitios web, redes sociales, aplicaciones… Finalmente, el desarrollador podrá comprobar los resultados con datos que se extraigan de los experimentos que realice con Badgeville en sus aplicaciones.

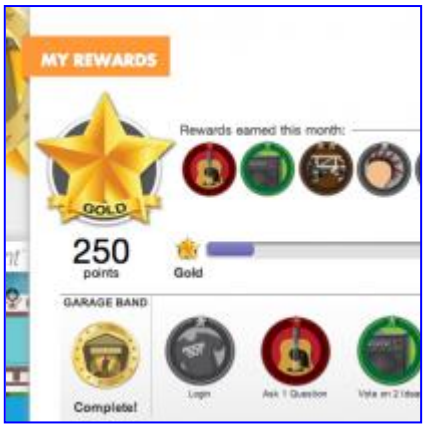

Ilustración 14: Mecánicas de gamificación con *Badgeville*
http://badgeville.com/sites/default/files/styles/207x207/public/contextual_rewards_0.png?itok=7wzkYek4

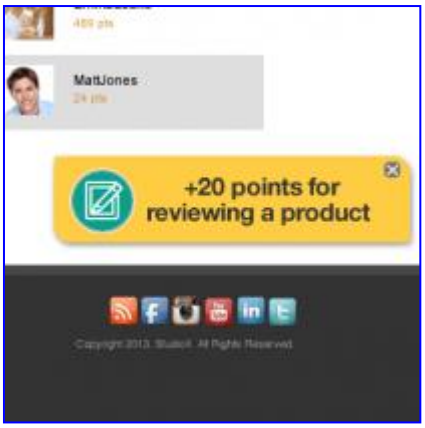

Ilustración 15: Notificación en *Badgeville*
http://badgeville.com/sites/default/files/styles/207x207/public/crouton_points.png?itok=J8afEcqU

Entre los elementos de gamificación que utiliza *Badgeville* de los que hemos visto en el apartado 2.6 Elementos de los juegos tenemos múltiples sistemas de puntuación, niveles de status, *leaderboards* contextualizados para el usuario, *badges* (Ver Ilustración 14), notificaciones en tiempo real (Ver Ilustración 15)… Todo ello se ofrece desde una visión de personalización y contexto, para que a cada usuario se le pueda motivar en los campos que se adecúen más a su relación con la aplicación o a su misma naturaleza.

4.2.2 Mozilla OpenBadges

Desde *Mozilla* se ofrece un programa que permite premiar los logros y el aprendizaje que se desarrolla hoy en día en las nuevas tecnologías. Debido a que no se otorgan premios tangibles por conocimientos adquiridos fuera de las aulas, *OpenBadges* nos ofrece una infraestructura digital que pretende solucionar este problema, para disponer de un reconocimiento por esas aptitudes que se desarrollan fuera de la escuela.

Para ello, se ofrece una gama de *badges* que se pueden customizar y contextualizar para las diferentes actividades que se quieran premiar, con una arquitectura que se puede descargar libremente en *GitHub*, en código libre, por lo que se podrá modificar y adaptar o mejorar dependiendo de las necesidades del desarrollador.

4.2.3 *Zurmo*

Zurmo es una herramienta útil para la relación entre clientes de una empresa o producto. Es una aplicación gamificada con una interfaz sencilla que pretende facilitar la vida a los jefes de ventas y los responsables de marketing.

A través de la plataforma de *Zurmo* se da un soporte de comunicación para las empresas a través de la gamificación, con *badges, leaderboards* y puntos. Todo el código es libre y puede ser sometido a cambios.

4.3 Userinfuser

Actualmente el software se puede encontrar en *Google Code*[17] junto a toda la documentación necesaria para su instalación y uso.

4.3.1 Arquitectura básica

El servicio que ofrece la arquitectura *Userinfuser* en cliente se basa en proporcionar al desarrollador elementos de gamificación y las funciones necesarias para gestionar los datos que los forman y obtener visuales en relación a esos datos y elementos. Así, podemos definir de forma general el funcionamiento de esta arquitectura con el gráfico básico de la ilustración 16.

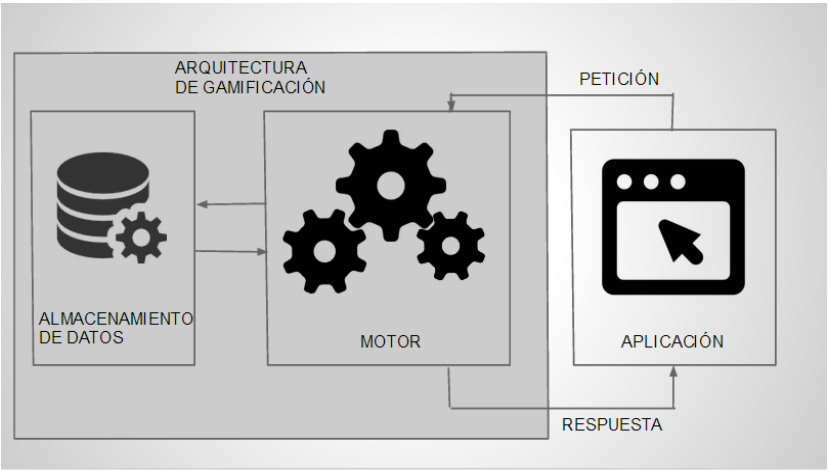

Ilustración 9 Diagrama básico de una aplicación con la arquitectura Userinfuser

[17] https://code.google.com/p/userinfuser/

Como se puede observar, la arquitectura *Userinfuser* es independiente de la aplicación, uniéndose a ésta mediante peticiones y respuestas que se gestionarán posteriormente. La arquitectura está compuesta básicamente de un motor y los datos correspondientes a los usuarios que utilicen la aplicación.

4.3.2 Motor de la arquitectura

El motor es el que hace funcionar a la arquitectura, se encarga de gestionar los datos para crearlos, modificarlos, borrarlos o simplemente para tener acceso a ellos. La arquitectura además tiene además su propio sistema de almacenamiento de datos, los correspondientes a la gamificación, como son los usuarios, los puntos y *badges* asociados a ellos… etc. Estos datos son los que se utilizarán en la aplicación para que mediante las acciones del usuario final se cambien y actualicen continuamente para dar sentido a la filosofía de gamificación. Por tanto, el control de la gamificación se hace mediante llamadas en cliente (aplicación) al servidor, con una serie de órdenes para que el motor en el servidor las lleve a cabo y las almacene. El motor de la arquitectura *Userinfuser* está programado en Python, lo que hace que el servidor que utilice la arquitectura soporte este tipo de lenguaje.

4.3.3 Almacenamiento de datos

El almacenamiento y la gestión de los datos se hace desde un servidor web que previamente hay que tener funcionando. Esto se puede hacer de dos formas: La primera, que es la más sencilla, es haciendo uso del servidor web de la plataforma *Cloudcaptive*[18]. En él se tiene todo lo necesario para no tener que preocuparnos de nada relacionado con la parte de servidor. Tanto el almacenaje de los datos como las funciones que los gestionan y modifican se

[18] https://cloudcaptive-userinfuser.appspot.com/adminconsole

encuentra implementado ahí y el desarrollador de aplicaciones únicamente tendrá que insertar las funciones necesarias en su código donde corresponda. Esta opción se ofrece desde la página web de *Userinfuser* como alternativa a la segunda: Crear y configurar un servidor web propio. Esta segunda opción se ofrece como alternativa a la primera y tiene varias ventajas. En primer lugar el desarrollador tiene control total sobre el servidor web ya que es suyo, por tanto podrá hacer las modificaciones que considerara oportunas en el código de servidor, del cual se dispone en *GitHub*[19] para su descarga totalmente gratuita, y adaptar así la arquitectura a sus propias necesidades y enfocar un código extenso aunque sencillo a optimizar las partes de las que va a hacer uso en su aplicación. Otra ventaja es que el servidor de *Cloudcaptive* no es gratuito, y habría que pagar una cuota mensual que el desarrollador se puede ahorrar. No obstante, configurar y mantener el servidor web propio no es una tarea trivial y puede que para el desarrollador sea mucho más cómodo no tener que preocuparse de ello. Desde el sitio web de *Userinfuser* en *Google Code* se ofrece la documentación necesaria para la implementar una plataforma de gamificación a través de un servidor propio[20]. Se basa en la configuración de un servidor a través de *Appscale*, un *framework* libre que soporta aplicaciones del motor *Google* [7].

4.3.4 Interfaz de Servidor

Sea cual sea la opción que se elija, el código proporcionado viene con una interfaz *PHP* para que la gestión pueda hacerse cómodamente sin necesidad de tener conocimientos de bases de datos (Ilustración 17).

[19] https://github.com/nlake44/UserInfuser

[20] https://code.google.com/p/userinfuser/wiki/Deployment

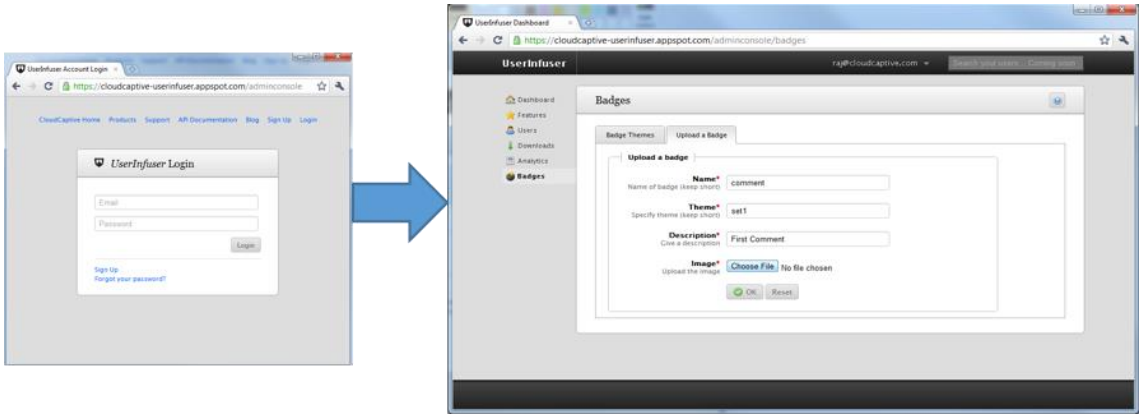

Ilustración 17 Login y creación de Badges

Imágenes obtenidas de:

Login: http://i.imgur.com/iyUY3.png

Creación de Badges: http://i.imgur.com/d37P2.png

Como podemos observar, lo primero que tenemos que hacer para entrar en la gestión web es identificarnos con un usuario y contraseña que debemos crear previamente. Este usuario y contraseña será único para cada aplicación que se quiera gamificar, ya que en la parte del servidor reservado cada usuario es donde estarán todos los datos de la gamificación de la aplicación que corresponde. Por tanto, el desarrollador tendrá tantas cuentas como aplicaciones con arquitectura *Userinfuser*, y podrá manejar los datos de todas desde el sitio *php* que se ha comentado antes, que en el caso de ser de *Cloudcaptive* se encontrará en su sitio web (https://cloudcaptive-userinfuser.appspot.com/adminconsole), y que en el caso de tenerla en nuestro propio servidor podremos verla en la dirección IP que hayamos configurado.

La información relativa a cómo utilizar esta interfaz, está explicada en su sitio en Google Code[21] con poca precisión, aunque es bastante intuitiva. Desde aquí podremos crear el esqueleto de nuestra gamificación ya personalizada para la aplicación correspondiente. Si nos fijamos en la parte izquierda del panel (Ilustración 18), podremos ver las opciones que se proporciona.

[21] https://code.google.com/p/userinfuser/

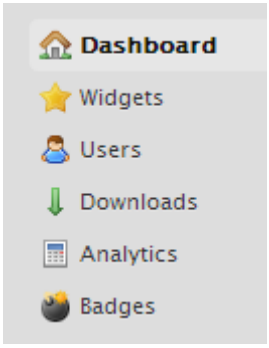

Ilustración 10 Menú de la Interfaz de Servidor
Imagen obtenida de: http://i.imgur.com/d37P2.png

Tenemos un apartado para los usuarios que nos dará la información de cada participante. Aunque dentro de esta pestaña se da la opción de crear y modificar usuarios, hay que tener claro que será desde la aplicación, en general, donde se crearán y modificarán los usuarios a través de las llamadas al servidor, ya que será el propio usuario final el que se cree a sí mismo desde la aplicación del desarrollador. Aun así, la interfaz nos proporciona esta opción para poder ver la información de cada uno de ellos y su progreso y si fuera necesario modificarlos independientemente de la aplicación. Los usuarios llevan asociada una puntuación que al igual que el usuario en sí no debería a priori ser modificada aquí.

Lo más interesante de esta interfaz es que nos permite crear los elementos de gamificación que vamos a usar. Suponiendo una aplicación muy sencilla en la que únicamente se pudieran obtener medallas (*badges*) cuando se alcanzaran unas puntuaciones determinadas, se crearían aquí esas medallas en el apartado *badges,* asignándoles una imagen y una razón por la que la hemos obtenido, quedando así registradas en nuestra aplicación y listas para que los usuarios finales pudieran conseguirlas mediante sus acciones. Dentro de la pestaña *widgets* se permite crear las características de los visuales que podamos necesitar, por ejemplo, un *leaderboard* basado en las puntuaciones de los usuarios. Desde aquí, se puede dar formato para que desde el código del desarrollador únicamente se haga una llamada a ese recurso y se devuelva para mostrarlo en la aplicación.

Por tanto, el uso de esta interfaz es básicamente el trabajo previo a la implantación de la gamificación en el código del desarrollador, es la creación del esqueleto del sistema que usaremos, personalizado para cada programa.

Como ya se ha mencionado, el código de esta parte en el caso de que utilicemos un servidor propio es totalmente modificable y adaptable a las necesidades de cada uno, pero siempre hemos de tener en cuenta que el servidor gestionará la gamificación de todos los programas que se hallen registrados en él, y por tanto es recomendable siempre adaptar este código con ampliaciones o mejoras de lo ya existente, ya que si se suprimen funciones que en un principio no se necesitan se puede perder funcionalidad quizá necesaria en futuras aplicaciones que se incorporen a este servidor.

4.3.5 Cliente

El código proporcionado por *Userinfuser* para la parte de cliente se encuentra programado en varios lenguajes, para poder dar servicio a multitud de aplicaciones. Podemos encontrar PHP, JAVA, Python y Rubi. En todos ellos las funciones tienen como tarea traducir las órdenes que se dan desde el programa que utiliza esta arquitectura para mandarlas al servidor y que se procesen y guarden. Este código es libre y aunque la documentación no es muy extensa no es necesario ahondar mucho en la comprensión del mismo para poder utilizarlo. Las funciones que ofrece sirven para otorgar puntos a un usuario, asignarle *badges* o *badge-points* conseguidos, petición de *widgets,* crear u obtener información de usuarios… Cuando utilicemos cualquiera de estas funciones, siempre obtendremos un mensaje de respuesta, tanto si es información que se utilizará después en la aplicación, como en la petición de un *widget* como si es simplemente un mensaje de confirmación de que la acción ha sido realizada con éxito. Debido a que la gestión de los datos desde el cliente se basa en peticiones al servidor, en las funciones que afecten a un usuario de la aplicación se tiene que especificar el nombre del usuario al que afectará esa función. Esto plantea una cuestión en la seguridad de los datos

que Userinfuser no resuelve, ya que asume que la privacidad de los perfiles se tratará en la aplicación, mediante un control de acceso mediante claves o *login* que verifiquen la identidad del usuario. Por tanto, una vez el usuario se haya identificado y autenticado en el programa, sus datos serán los que se utilicen para hacer las peticiones al servidor.

4.3.6 Funciones

A continuación se detallarán las funciones de las que se dispone en la arquitectura *Userinfuser*.

4.3.6.1 Manejo de Usuarios

En primer lugar, encontramos las funciones asociadas a la gestión de los usuarios. Userinfuser nos ofrece la posibilidad de crear y modificar usuarios, mediante la función *update_user.* Como parámetros obligatorios está su ID, único para cada usuario, aunque hay posibilidad de dar más información, como un nombre de usuario, un link a un perfil externo y un link a una imagen de perfil, para incluirla en la aplicación.

Por otro lado tenemos la función que nos devuelve la información del usuario, llamada *getUserInfo,* que devuelve un *String* con la información del usuario codificada como aparece en la Tabla 1, según la documentación del código.

```
<code>
        * {"status": "success",<br/>
        * &amp;nbsp;&amp;nbsp;&amp;nbsp;"is_enabled": "yes",<br/>
        * &amp;nbsp;&amp;nbsp;&amp;nbsp;"points": 200,<br/>
        * &amp;nbsp;&amp;nbsp;&amp;nbsp;"user_id": "nlake44@gmail.com",<br/>
        * &amp;nbsp;&amp;nbsp;&amp;nbsp;"badges":  ["muzaktheme-guitar-private",<br/>
        * &amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;"muzaktheme-bass-private",<br/>
```

```
* &amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;"muzaktheme-drums-private"],<br/>
* &amp;nbsp;&amp;nbsp;&amp;nbsp;"profile_img": "http://test.com/images/raj.png",<br/>
* &amp;nbsp;&amp;nbsp;&amp;nbsp;"profile_name": "Raj Chohan",<br/>
* &amp;nbsp;&amp;nbsp;&amp;nbsp;"profile_link": "http://test.com/nlake44",<br/>
* &amp;nbsp;&amp;nbsp;&amp;nbsp;"creation_date": "2011-02-26"}<br/>
* </code>
```

Tabla 1

Debido a esta forma de acceder a los datos de usuario, el programador de la aplicación deberá ser capaz de decodificar esta información y almacenarla mediante código programado en el caso de que necesitara esta información para algo concreto; la respuesta de esta función está pensada para utilizarse como código embebido y que se añada directamente a nuestro código HTML. El procedimiento se puede ver gráficamente en la Ilustración 19.

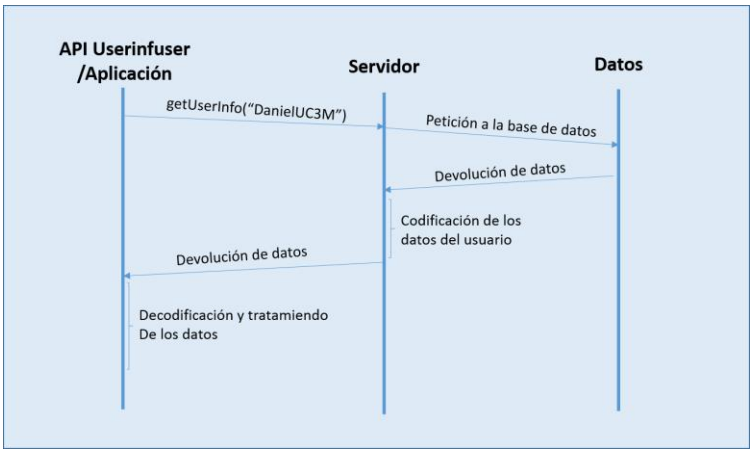

Ilustración 19: Diagrama de petición de la información de usuario mediante la función "getUserInfo" en Userinfuser.

4.3.6.2 Manejo de puntos y *badges*

La función que se encarga de dar puntos al usuario es *award_points* y en ella tenemos que especificar cuantos puntos queremos darle al usuario y la razón por la cual ha ganado esos puntos. Este último parámetro es opcional y no tiene porqué ser rellenado en todas las ocasiones.

Por otro lado, *Award_badge* es la que asigna el premio conseguido especificado en los parámetros de la función. En este caso *Userinfuser* opta

por mandar una cadena de caracteres en las que se codifica el *badge* conseguido mediante 2 valores concatenados con guiones: *temática-nombre*. Esto se decodifica en el servidor y se hacen los cambios oportunos. La temática y el nombre que se manden en la función deben existir previamente en el servidor, que habrán sido introducidos mediante la interfaz anteriormente explicada. Otras opciones que ofrece esta función y que no son obligatorias son añadir la razón de entrega de ese premio y un *link* que se usará para redirigir al usuario final si hace *click* en la imagen del *badge*.

Además, otra opción que se ofrece para los *badges* son los llamados *badge-points* que ofrecen la posibilidad de dar puntos asociados a un premio, que se desbloqueará cuando consigamos los puntos que se necesitan. En este caso, deberemos informar, aparte de la información anterior, cuántos puntos se le otorgan al premio de este usuario y cuántos puntos es el objetivo a alcanzar. En el caso de que estos dos últimos datos coincidan, el premio se desbloqueará como conseguido.

Por último, se incluye también una función *remove_badge*, que como su nombre indica, quitará el premio especificado al usuario especificados en los parámetros.

4.3.6.3 Funciones para recursos visuales

Las acciones que realizan las funciones que se han explicado hasta ahora son totalmente invisibles para el usuario, ya que se basan en modificar datos en el servidor y en algunos casos el propio programador no querrá que el usuario final sea consciente de estos cambios. Sin embargo, para que la gamificación pueda ser apreciada y aprovechada por el usuario, se ofrece la función *get_widget*, que devuelve del servidor lo necesario para mostrar visualizaciones

en la aplicación. En este apartado Userinfuser nos ofrece *iframes*[22], que se devuelven en forma de cadena de caracteres para insertarlos en el código de forma dinámica. Los *iframes* de los que dispone Userinfuser son:

Trophy-case: Todos los *badges* conseguidos por el usuario.

Leaderboard: Tabla comparativa con los usuarios.

Points: Los puntos que tiene el usuario.

Rank: La posición global en la que se encuentra el usuario dentro de la aplicación.

Notifier: Si queremos visualizar cambios en tiempo real, como premios obtenidos o puntos conseguidos.

Milestones: Para los *badge-points*.

Todos los anteriores *iframes* que obtenemos no son fácilmente modificables por la aplicación, ya que como se ha dicho vienen en forma de cadena de caracteres que introduciremos en el código. La personalización de estos *widgets* se llevará a cabo en la interfaz del servidor, que nos permite editar todos los parámetros de los que constan los *iframes* (Ilustración 20).

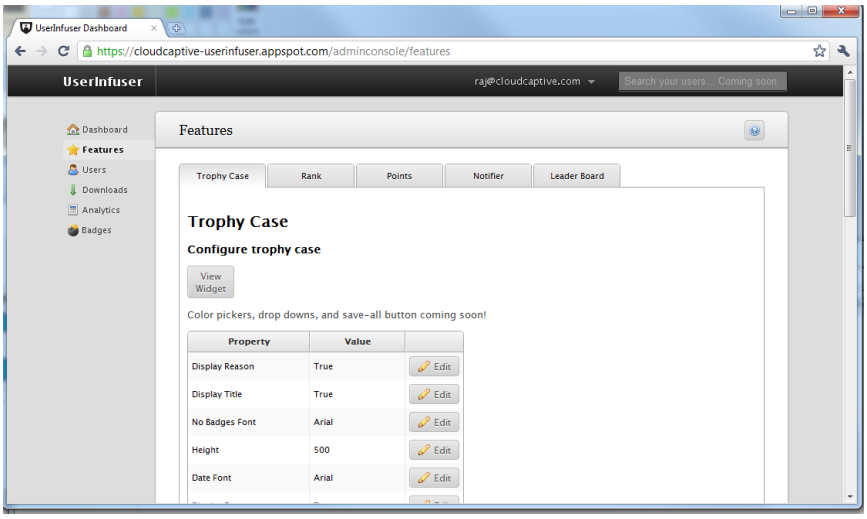

Ilustración 20: Ventana de edición de Widgets en Userinfuser
Imagen obtenida de: http://i.imgur.com/Ot0He.png

[22] Un iframe es un marco que nos permite embeber contenido en una aplicación web. [8]

El procedimiento se describe gráficamente en la Ilustración 21:

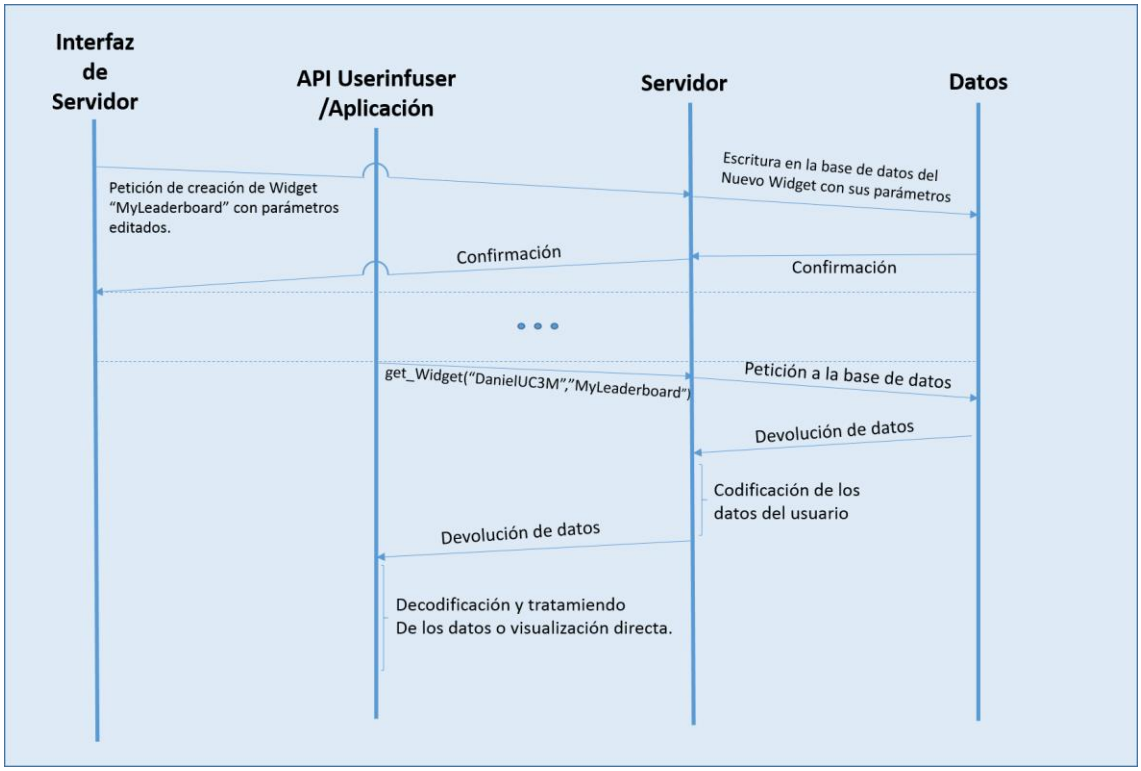

Ilustración 21: Diagrama de creación y petición de Widget con Userinfuser

4.3.7 Uso funcional y valoración

La arquitectura *Userinfuser* ofrece un servicio que para la mayoría de las aplicaciones será suficiente. Aunque su funcionalidad no es completa, es una propuesta interesante para la aplicación de la gamificación. Por supuesto, y atendiendo a los requisitos que debe tener un sistema de gamificación para conseguir los objetivos para los que se aplica (ver apartado 4.4), no será suficiente. Un buen sistema gamificado debe tener dinámicas y mecánicas que lo hagan funcionar correctamente, y eso no se ofrece en esta arquitectura. Aplicar bien un *framework* de Gamificación dependerá del desarrollador de la aplicación y la estructura de juego que establezca, es decir, del uso funcional que le dé a la arquitectura.

4.4 Propuestas de mejora de Userinfuser

Tras establecer en el Capítulo 3 los requisitos que tiene que tener una arquitectura de gamificación, estamos en disposición de hacer un análisis crítico de la arquitectura *Userinfuser* y proponer cambios y mejoras en su implementación para que pueda ampliar su funcionalidad y así ser más útil y versátil para las aplicaciones que hagan uso de ella.

4.4.1 Cliente

Dentro de la parte de cliente la mejora que podría aplicarse a *Userinfuser* es, en primer lugar, los lenguajes ofrecidos para su implementación. En la propia descarga de la API nos encontramos que disponemos de las herramientas de clientes programadas para *PHP, Python, Rubi* y *Java*. Aunque la oferta no es escasa, parece que se pretende dar un servicio de gamificación con *Userinfuser* a plataformas web, sin embargo, sería conveniente tener una versión móvil para poder ampliar la funcionalidad y alcance, ya que cerca de 800 millones de personas usan ya Android [4]. La "traducción" a ese lenguaje tampoco sería una tarea muy tediosa debido a la similitud entre java y Android. Esto daría la posibilidad a los clientes potenciales de esta arquitectura de desarrollar sus servicios en este tipo de plataformas sin tener que buscar otro sistema de gamificación o incluso renunciar a él.

4.4.2 Utilidades

Entre las mejoras que se pueden aplicar a la funcionalidad de Userinfuser está añadir algunos elementos de gamificación que se echan en falta.

En general, la versión actual de Userinfuser no ofrece posibilidad de incluir misiones, niveles de acceso o grafo social, por lo que se presenta como una arquitectura básica que se centra en ofrecer la ya explicada triada PBL (ver 2.8 La triada PBL).

Además, no es posible subir de nivel, con lo cual la puntuación es absoluta, perdiendo la sensación de progreso que se podría tener. Para implementar esta mejora de una forma sencilla podría comprobarse la puntuación total cada vez que se ganan puntos dentro de la propia función *awardPoints* y compararla con unos números editables por el programador, de forma que si se superan alguno de los puntos establecidos se llamará a una nueva función *levelUp* que manda una orden al servidor para que suba un nivel al usuario. Por otro lado, se necesitaría incluir en la base de datos un campo "*level*" para los usuarios.

4.4.3 Funciones

Entre las mejoras que se podrían aplicar a las funciones de Userinfuser, lo más importante es la forma de tratar los datos. Como veíamos en la Tabla 1, los datos recibidos por parte del servidor había que o bien decodificarlos en cliente o mostrarlos tal cual en la aplicación, ya que venían en formato HTML. La propuesta de mejora consiste en poder acceder a los datos de forma independiente, para así poder hacer un uso de las respuestas más directo y menos complejo. Para ello, en la función *getUserInfo* se debería poder especificar qué información queremos que se devuelva y que desde el servidor la respuesta sea únicamente los datos solicitados. Este cambio en la función quitaría una funcionalidad que se pretende dar desde Userinfuser y es la

impresión directa de toda la información conjunta, pero en consecuencia se ganará considerablemente en libertad de edición y dado que la respuesta tampoco tiene un formato muy complejo no es una pérdida importante.

Para la funcion *awardBadgePoints*, se necesita incluir como parámetros los puntos que se añaden así como los requeridos para ganar el *Badge*. Si mejoramos la base de datos para que a partir del nombre del *Badge* el servidor sepa cuántos puntos hacen falta para desbloquarlo y actúe en consecuencia, ganaremos eficacia en cliente y no será necesario el envío de datos redundantes que se pueden evitar.

Por último, en el código de cliente se ofrece también una función para solicitar *Widgets* o visuales, pero en el caso de las notificaciones no es deseable tener que solicitarlas para que realicen, sino que aparezcan automáticamente cuando un usuario alcance algún objetivo. Además, si se tiene que hacer una petición cada vez que deba aparecer una notificación, esto supone que el programador de aplicaciones tendrá que comprobar cada vez que haga una llamada si es momento de presentar una notificación, con lo cual se le está dando una carga innecesaria que se podría evitar.

Capítulo 5: Propuesta de Arquitectura

5.1 Introducción

En este capítulo se presentará una propuesta de una arquitectura de gamificación para **dispositivos móviles**, describiendo las partes de la misma y ahondando un poco más en las funciones y características que debe tener para cubrir los requisitos que se presentaban en el apartado 3.4 Requisitos de una arquitectura de gamificación. Se presentará a continuación empezando desde una visión general del sistema hasta describir detalladamente los cometidos de cada una de las partes que la forman, para dar una visión detallada de la funcionalidad y una descripción técnica de su implementación.

5.2 Diagrama básico

La arquitectura la formarán 3 partes diferenciadas (ver Ilustración 22):

- Base de datos de servidor.
- API de cliente.
- Servidor (Motor).

En primer lugar, el servidor que se propone es un servidor WAMP con Apache, ya que el manejo es bastante sencillo y permitirá tener más organizadas las distintas aplicaciones que desarrollemos. En ese servidor, almacenaremos los archivos que contienen el código que maneja la gamificación, esto es, el motor. El lenguaje que parece más manejable y versátil para el servidor es PHP, teniendo en cuenta que las sucesivas llamadas a las funciones se harán a través de la red y este lenguaje es el más apropiado para manejar esas llamadas con los métodos get y post[16].

Por otro lado, la base de datos, que se encuentra en el servidor, estará escrita en MySQL. Esta elección se basa en la sencillez del código MySQL, que permite accesos y modificaciones en pocas líneas de código, quitando carga computacional al servidor.

Por último, la API de cliente serán las librerías necesarias para que el programador de la aplicación pueda hacer uso de la arquitectura de forma que no tenga que conocer su funcionamiento, sino simplemente insertando las funciones de las que dispondrá instalando este complemento.

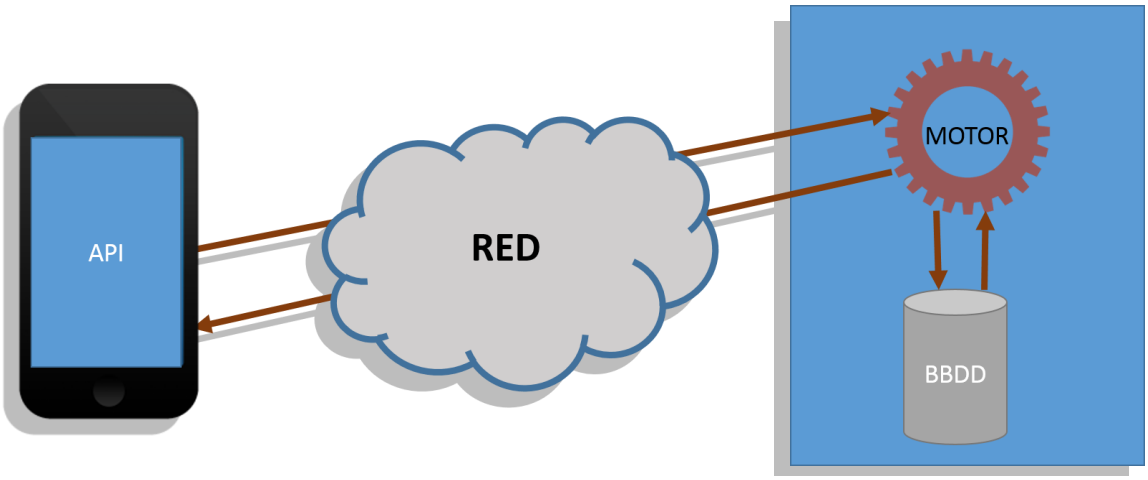

Ilustración 22: Diagrama básico de la arquitectura móvil propuesta

El funcionamiento básico de esta arquitectura es el siguiente:

Desde la aplicación se mandarán una serie de llamadas que pasarán por la red llegando al motor de la arquitectura situado en los archivos PHP que nombrábamos anteriormente. Los archivos PHP de los que se dispondrá serán uno para cada acción que necesitemos. Por lo cual cada llamada desde la API estará asociada a un archivo PHP que hará las acciones correspondientes. Esas acciones serán siempre de acceso a la base de datos, unas veces para coger información y otras para modificarla.

5.3 Base de datos

En primer lugar, para almacenar los datos de la gamificación se usará una base de datos *MySQL*, como se ha explicado debido a su facilidad de acceso y su poca complejidad. Esta base de datos debe contener toda la información asociada a los datos de los usuarios que utilicen la aplicación así como los componentes (ver apartado 2.6.3 Componentes) de la gamificación. La organización de los datos es importante ya que de ella dependerá toda la filosofía de la gamificación. En este caso, se organizarán de tal y como se explica a continuación.

5.3.1 Usuarios

Para almacenar los datos de usuario se tendrán varias tablas que contengan toda su información relativa a la gamificación. La primera de estas tablas será la tabla **Usuario**, que contendrá los siguientes campos:

- ID: Un nombre único para cada usuario que lo identificará de los demás y que se utilizará siempre que queramos referirnos a alguien concreto. Esta identificación no podrá repetirse en esta tabla y será clave en todos

los procesos de acceso a la base de datos, ya que siempre será necesario referenciar las llamadas al usuario que realiza las acciones.

- Link de Avatar: Una ruta de acceso a la imagen que representa al usuario. Como la arquitectura está pensada para aplicaciones móviles, esta ruta podrá ser interna del dispositivo o también un enlace a una dirección web. Se almacenará como formato texto, ya que la tarea de descifrar el link y acceder a su contenido la llevará a cabo la propia aplicación.

- Contraseña: Para que el acceso únicamente pueda hacerlo el propietario del perfil.

- Nivel: Un número, que por defecto será 1.

- Progreso de nivel: Un número que representará el porcentaje del nivel actual del usuario.

- Puntos: Los puntos totales que se asocian a ese usuario

- Nivel de acceso: Un número que se utilizará para proporcionar un acceso determinado al contenido de la aplicación, que tendrá que gestionar el programador de aplicaciones.

Como se puede observar, todos los campos que contiene esta tabla hacen referencia a información que no es variable en cuanto a volumen de datos, es decir, siempre sabremos cuánto espacio de almacenamiento necesitamos para almacenarla. Por ejemplo, los niveles sabemos que siempre será un entero así como el link de Avatar será una cadena de caracteres que podremos limitar su longitud.

La siguiente tabla que necesitamos es la tabla de **Información de Usuario,** con los siguientes campos:

- ID: Que sería la misma identificación que en la tabla anterior, con la diferencia de que el valor que contenga sí podrá repetirse en las

distintas entradas de la tabla. Esta identificación se usará para saber a qué usuario corresponde la información de esta entrada, y debido a que un usuario puede tener diferentes campos de información, dependiendo de la que se quiera incluir en la aplicación (Ciudad, Población, E-mail, Departamento, Asociación...), habrá tantas filas con un mismo ID como unidades de información se hayan incluido.

- Título: Hace referencia a una pequeña descripción de la información que está almacenada en esta fila. Por ejemplo, “departamento”.

- Valor: En él se albergará el dato correspondiente al título y la identificación de esa fila. En el ejemplo de “departamento”, un valor que podría contener sería “Contabilidad”.

Así, en esta tabla se incluirán todos los datos adicionales que no estaban incluidos en la tabla “usuarios”. En esta tabla la cantidad de información es variable, ya que dependerá de cuantos datos quiere la aplicación que almacenemos. Unas veces no será necesario almacenar nada en esta tabla, si no es relevante más información, y otras veces tendremos gran cantidad de campos en esta tabla, si se quiere que los perfiles de la aplicación sean más completos en cuanto a datos personales de usuario.

La forma en la que accederemos a los datos de esta tabla siempre vendrá dado por el ID de usuario (ver Ilustración 23). Si accedemos a esta tabla buscando todos los campos cuyo ID sea uno en concreto, la respuesta será toda la información relativa a ese usuario.

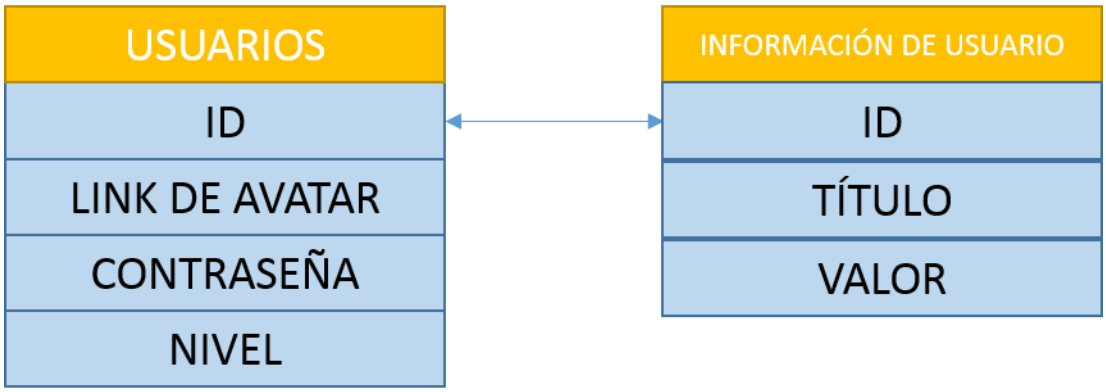

Ilustración 23 Conexión entre las tablas Usuarios e Información de Usuario

Por otro lado, los usuarios también tendrán que tener almacenado en la base de datos qué premios han conseguido. Esta será la tabla **Premios de usuario,** y tendrá los siguientes campos:

- ID: Al igual que en la tabla “Información de usuario”, podrá estar repetido tantas veces como badges haya conseguido un usuario. Será el punto de unión con la tabla “usuarios”.

- Tipo: Para definir qué tipo de premio es el que se quiere almacenar como ganado. En nuestro caso este campo podrá valores como “*Badge*” si es una medalla, “*Trophy*” si es un trofeo, “*BadgePoint*” si es una medalla desbloqueable por puntos… etc.

- Nombre: Define el título que se le ha dado al premio en cuestión. Por tanto, los campos clave en esta tabla serán conjuntamente “Tipo” y “Nombre”, ya que nunca podrá haber dos premios diferentes que tengan el mismo nombre y sean del mismo tipo. Los valores que podamos poner tanto en “Tipo” como en “Nombre” no serán cualesquiera, sino que estarán definidos en otras tablas que se explicarán más adelante.

- Puntuación actual: Los puntos asociados a este premio que tiene actualmente el usuario. Si la puntuación requerida para el premio es 1, este campo únicamente puede tener valor 1, ya que si se ha actualizado este premio únicamente puede ser para conseguirlo.

También es necesaria una tabla que contenga los diferentes equipos a los que puede estar asociado un usuario, ya que un usuario puede estar suscrito a varios equipos. Los campos serían:

- ID: Que representa al usuario.
- Nombre: Que identifica el equipo al que pertenece.

Por último, hay que incluir en la base de datos la tabla correspondiente al grafo social. En este caso, la forma más sencilla es incluir los ID de los usuarios con los que se tenga relación, para crear una red social sencilla:

- ID
- ID conectado: Que representa al perfil de la persona con la que el usuario está conectado.
- Vínculo: ID de la persona que hace que el usuario esté conectado a ese usuario. En caso de que el vínculo sea directo, este campo será el mismo que ID.

Una vez hemos visto estas primeras tablas, ya se puede intuir cómo será la estructura de almacenamiento de datos de los usuarios. Partiendo del valor que identifica a éste, su ID, podremos acceder a toda la información que contiene así como añadir y quitar campos sin interferir con los de otros usuarios.

5.3.2 Premios

La necesidad de tablas que almacenen los premios surge debido a que todos los premios que se otorguen dentro de la aplicación gamificada estarán previamente definidos. Los premios no se generan dentro del programa en el

que esté implantada la arquitectura, sino que existirán con anterioridad en la base de datos y la aplicación únicamente accederá a ellos para mostrarlos o para asignarlos a un usuario. Por tanto, la creación y edición de los premios es una tarea que ha de ser realizada por el cliente de forma previa al uso de las funciones. Cuando un programador de aplicaciones tenga la intención de hacer uso de esta arquitectura de gamificación, deberá pensar, entre otras cosas, el sistema de premios que quiere conceder a sus usuarios. Una vez tenga este sistema estructurado de forma teórica, deberá incluir estos premios en la base de datos para posteriormente programar las acciones en la aplicación a través de la API, ya que no se podrá premiar nada que no esté definido previamente en la base de datos.

De esta forma, los premios que se podrán conceder a través de esta arquitectura serán: Badges, BadgePoints, Trophies y Milestones.

5.3.2.1 Badges y Badgepoints

Estos premios serán los básicos que se concederán dentro de la aplicación, aunque por supuesto esto depende de la estructura que aplique el programador. Visualmente vendrán definidos por una imagen típicamente como las que veíamos en la Ilustración 6 (Capítulo 2) y que normalmente tendrá coherencia temática con la razón por la cual se otorgan esos premios. Los campos de los que dispondrán estas tablas serán:

- Nombre: Único tanto para la tabla *Badges* como para *BadgePoints*. Es el campo que identifica el componente.

- Temática: Para agrupar los *badges* en clases que servirán para poder otorgar posteriormente los trofeos de coleccionista. Cada temática tendrá un número determinado de *badges* y está pensado para organizar los premios en categorías que tengan ciertas características en común.

- Imagen: Que contendrá la ruta a la imagen que identifica el componente. Al igual que con el avatar del perfil de usuario, esta imagen podrá contenerse en el propio dispositivo, como contenido descargable junto con la aplicación. De esta forma, no será necesario acceder a la red para descargarse la imagen identificativa del premio.

- Descripción: Una cadena de caracteres en la que se especifica el motivo por el que se otorga el premio o los objetivos que hay que conseguir para desbloquearlo.

Además en el caso de los BadgePoints habrá otro campo que será **puntuación,** que consiste en un entero que establece el número de puntos necesarios para desbloquear el BadgePoint.

5.3.2.2 Trophies

Los *trophies* son un tipo de premio que calificará la progresión del usuario teniendo en cuenta los *badges* que haya conseguido. A partir del campo "temática" que se había definido en el punto anterior, cuando se consigan todos los *badges* asociados a una temática determinada se concederá un trofeo. Asi, a modo de ejemplo, si un usuario ha desbloqueado todos los *badges* asociados a comentarios de artículos en una aplicación tipo WIKI (ver apartado 3.2 Escenario 1: WIKI Corporativa), ganará el trofeo "Comentarista".

Los campos de esta tabla serán:

- Temática: Que hace las funciones del nombre del apartado 6.2.2.1. En este caso, la temática define al trofeo, ya que habrá un trofeo por cada temática que cree el programador de aplicaciones.

- Imagen: Igual que en el apartado 6.2.2.1

- Descripción: Igual que en el apartado 6.2.2.1

- Número de *badges* asociados: El número de premios de esa temática que hay que conseguir para desbloquearlo. Dado que los premios son únicos, cada vez que se gane uno con la temática de este trofeo se añadirá una unidad en el campo “puntuación actual” en la tabla “premios de usuario” y en la fila correspondiente al trofeo que tenga esa temática. Cuando la “puntuación actual” del usuario coincida con el campo “Número de *badges* asociados”, se desbloqueará el trofeo (Ilustración 24)

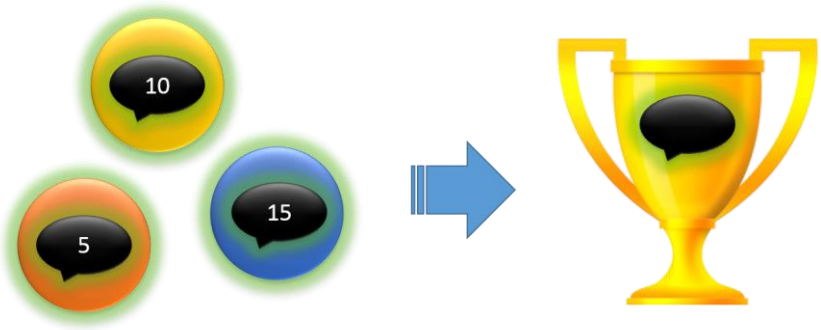

Ilustración 24: *Badges* y trofeo temático asociado.

5.3.2.3 Milestones

Los *Milestones* se pueden asemejar a “misiones” que el usuario deberá hacer para desbloquearlos. A diferencia de los trofeos o los *badges*, estos premios siempre serán visibles para que el usuario pueda saber cómo conseguirlos. Constan de tres objetivos que el usuario deberá conseguir para poder desbloquearlos. Sus campos son:

- Nombre
- Objetivo primero: Descripción del primero objetivo
- Objetivo segundo: Descripción del segundo objetivo.
- Objetivo tercero: Descripción del tercer objetivo.

- Imagen

- Descripción general.

Cada objetivo de los *Milestones* tendrá asociada una puntuación, 1, 2 y 4 puntos respectivamente. Si el usuario consigue el primer objetivo de un *Milestone*, en el campo "puntuación actual" de la fila correspondiente a ese premio de la tabla "Premios de usuario" se añadirá un punto, 2 si consigue el segundo y 4 si consigue el tercero. Así siempre podremos identificar qué objetivos ha conseguido, ya que si consigue el primero y el tercero la puntuación será de 5, que sólo puede estar formada por 1+4. El premio se desbloqueará cuando la puntuación de ese premio llegue a 7 (1+2+4).

5.3.3 Diagrama final de la BBDD

Una vez vistas todas las tablas que tiene la base de datos de la arquitectura de gamificación, el diagrama general que resulta es el que se puede observar en la Ilustración 25.

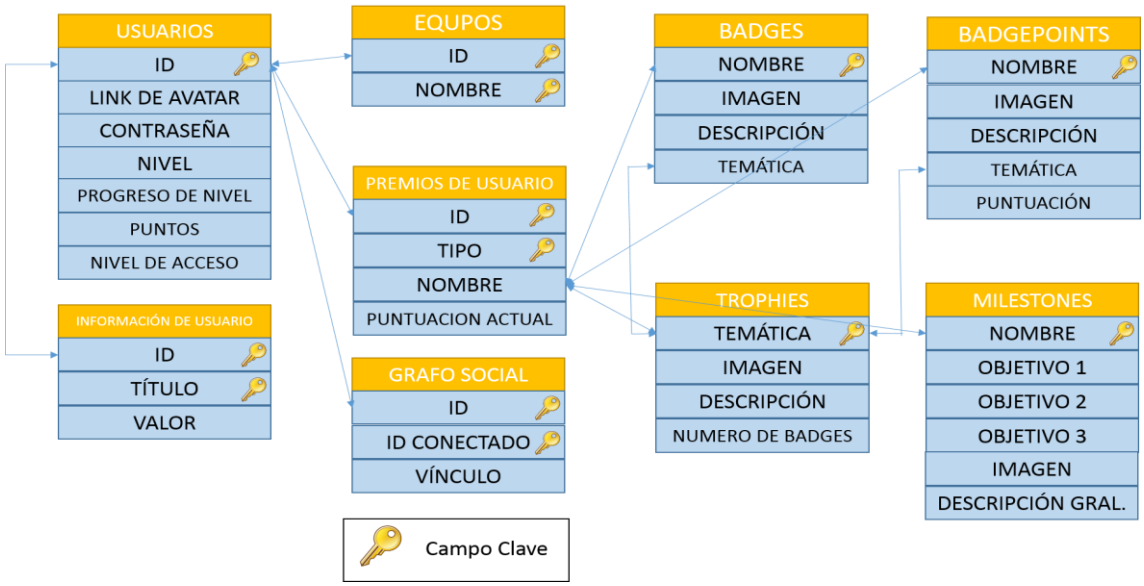

Ilustración 11: Diagrama completo de la Base de Datos

Los campos clave corresponden a los que definen por si solos un elemento de la base de datos. A un usuario, por ejemplo, le define su ID. En el caso de los premios de usuario e información de usuario, se definen por 2 campos, que forman por tanto una pareja que no se podrá repetir en ningún otro elemento de la tabla correspondiente.

5.4 API de Cliente

Como se explicaba anteriormente, la API de Cliente contiene las funciones necesarias para hacer todas las llamadas a la parte de servidor desde la aplicación de Cliente, en este caso la aplicación móvil del desarrollador que utilice la arquitectura. Se usará como un complemento para hacer uso del sistema de gamificación de forma paralela a la funcionalidad propia de la aplicación.

Las funciones que debería tener la arquitectura están definidas en el apartado 3.4.3 Funciones, y se explicarán detalladamente en los siguientes apartados.

5.4.1 Funciones de usuario

En primer lugar, las funciones que harán gestión de los usuarios serán **crear, actualizar y borrar usuario.**

Cuando desde la aplicación se quiera crear un usuario, a través probablemente de un formulario como el de la Ilustración 26, se tendrá que llamar a la función *crearUsuario.* Esta función tendrá como parámetros el ID y la contraseña que hayamos introducido. Para crear el usuario no es necesario nada más, ya que esa es la única información que se pedirá al usuario cuando quiera acceder a su cuenta. Esa información se encapsulará en parámetros en la aplicación y se enviará mediante POST a la dirección web del archivo PHP que se encuentra

en el servidor (ver Tabla 2). Una vez se crea el usuario y se accede a la aplicación, lo más sensato para el programador es que una variable global contenga el ID de usuario y así pueda accederse a ella en todo momento.

Después se podrá añadir más información mediante la función *actualizarUsuario.* Esta función tendrá como parámetros el ID de usuario, el campo que se quiere actualizar y el valor. Esos datos también se encapsularán en una variable de parámetros y se enviarán al correspondiente PHP en servidor.

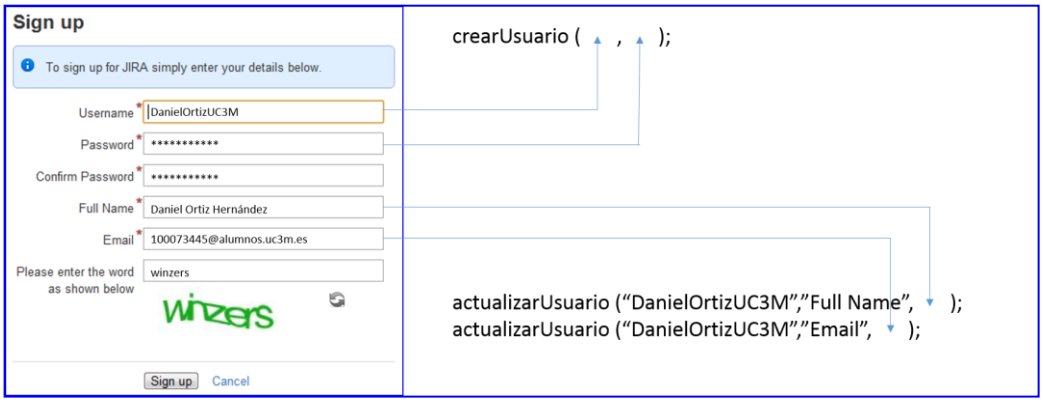

Ilustración 26: Formulario de registro
https://confluence.atlassian.com/download/attachments/185729483/captcha.png?version=4&modificationDate=1351471826458&api=v2

Como se dijo al final del apartado 5.2 Diagrama básico, cada función tendrá su archivo PHP asociado y así se tendrá una mayor distribución ordenada de la funcionalidad, más accesible por parte del programador, por lo que la dirección de envío será también diferente dependiendo de la función que haga el envío.

```
List<NameValuePair> params = new ArrayList<NameValuePair>();
    params.add(new BasicNameValuePair("User","DanielOrtizUC3M"));
    params.add(new BasicNameValuePair("Password","contrasenya"));
```

Tabla 2: Encapsulación de parámetros a enviar por POST en Android

Por último, la función *borrarUsuario*, nos permitirá a través de un solo parámetro (el ID) hacer una llamada al PHP en el servidor que borre el registro de la Base de datos MySQL.

Todas las anteriores funciones son de escritura en la base de datos. En teoría no se necesitaría información de respuesta, pero dado que estamos utilizando conexiones a través de la red, y es posible que la conexión entre cliente y servidor no se produzcan satisfactoriamente, es necesario que desde el servidor se envíe una respuesta asociada al éxito de la orden. En este caso, una confirmación tipo “0” si todo se ha realizado correctamente sería suficiente. En caso contrario, se enviará un “-1”. En el supuesto de que haya un error porque el registro está duplicado (hay que recordar que no se pueden repetir los campos clave de la base de datos) el resultado será un “1”.

Además, se necesita también una función para consultar los datos de un usuario. En este caso sería muy simple ya que con una función ***infoUsuario*** que a partir de un ID mande una petición al servidor para que acceda a la base de datos y nos devuelva la información relativa a esa identificación sería suficiente.

5.4.2 Funciones de puntuación y premios

Las funciones relativas a los puntos y premios de que dispondrá esta arquitectura son en relación a los visuales que se mostrarán al usuario en la aplicación para mostrarle su progreso dentro del sistema de gamificación.

En primer lugar, la función asociada a puntuar sería muy simple, ya que únicamente a partir de un número de puntos que el programador asociaría a una acción en una parte del código de su aplicación, se mandaría esa cantidad encapsulada al servidor para que éste acceda a la base de datos y aumente la puntuación con el número dado. El valor que se devuelva desde el servidor será la puntuación total del usuario con los puntos ya sumados, y esa cantidad se gestionará en la función para saber si se ha subido o no de nivel. En caso

de que así sea, se procederá a llamar a la función que mostrará el visual correspondiente y que se explicará más adelante.

En el caso de los *badges*, los parámetros que deberemos incluir en la función que los maneje sería el ID del usuario y el nombre del *badge*. Si atendemos a la estructura dentro de la base de datos de los premios de usuario, tendremos que especificar además qué tipo de premio estamos concediendo y qué puntuación actual asociada tiene. En el caso de los *badges* esto no será de gran complejidad ya que sólo pueden tener puntuación 1, debido a la propia naturaleza de los *badges*: O se ganan completos o no se ganan. Para el caso de los *Trophies, BadgePoints* y *Milestones* el proceso es el mismo con la diferencia de que la puntuación será la que corresponda en cada caso.

Las funciones de puntos, *badges*, *Trophies*, *BadgePoints* y *Milestones* tendrán como valor de retorno el éxito o fracaso de la llamada al servidor, pero además un tercer código que asociaremos a haber ganado un premio (incluyendo de forma implícita el éxito de la llamada al servidor). En el caso de los *badges*, los premios se ganarán siempre que se llame a esta función por tanto la respuesta desde el servidor siempre será este tercer código (que podemos llamar '1'). Para los demás, dependerá si la puntuación que hemos asociado a la llamada da como consecuencia haber llegado al objetivo de puntuación para ganar ese elemento o, en el caso de los puntos, haber llegado a la puntuación necesaria para subir de nivel.

5.4.3 Visuales

La gestión de los visuales era un tema que ya se había planteado en el apartado 3.4.2.2 Visualización como algo que debería crear y personalizar el programador de la aplicación, para así tener más libertad en cuanto a estética. Esta solución es muy versátil pero supone un trabajo añadido para el programador, por tanto, desde la arquitectura se dispondrá de recursos visuales por defecto que el programador podrá utilizar en su aplicación si lo desea.

De esta forma, las funciones que se ofrecerán desde la arquitectura serán llamadas de forma automática, en los supuestos que veíamos en el capítulo anterior. Es decir, los visuales aparecerán cuando haya un acontecimiento dentro del sistema de gamificación que requieran una muestra gráfica, como ganar un premio, subir de nivel…

Las funciones propuestas, una para cada tipo de visual que se quiera mostrar, tendrán como parámetros la identificación del elemento que se ha desbloqueado. Esta información se mandará al servidor, que consultará en la base de datos de los elementos y devolverá toda la información relativa a él. Ya de vuelta en la función, y con todos los datos disponibles, se creará un visual por defecto que consistirá en una pantalla y una notificación para informar al usuario, como se muestra en la Ilustración 27.

Ilustración 12 Notificación de *badge*

Por otro lado, para los *leaderboards* será suficiente con hacer una petición al servidor especificando qué tipo de datos necesita para ese *leaderboard* (relativos a un equipo generales… etc) y que éste devuelva una lista de usuarios ordenados para que se presenten después en la aplicación con la edición que elija el programador.

5.4.4 Resto de funciones

Para las funciones que quedan, como el grafo social o los niveles de acceso, los parámetros que se considerarán en estas funciones son los que se requieren en la base de datos, para incluir la información. Las llamadas se harán de forma similar a las funciones anteriormente explicadas, con las respuestas asociadas al éxito o fracaso de la conexión. La gestión de esas llamadas y las correspondientes respuestas correrá a cargo del programador de aplicaciones.

Capítulo 6: Conclusiones y trabajo futuro

Una vez planteado el proyecto estamos en disposición de extraer las conclusiones finales y marcar las líneas del trabajo futuro, dando así el proyecto por concluido.

6.1 Conclusiones

El desarrollo del presente proyecto me ha servido personalmente de aprendizaje de una técnica que abre y da gran versatilidad a los conocimientos que he ido adquiriendo a lo largo de los años de universidad.

En principio el proyecto pretendía ofrecer además una implementación de la arquitectura de gamificación propuesta, pero finalmente se tuvo que reducir el trabajo por falta de tiempo y dejar la propuesta en un diseño en profundidad que marcara un trabajo futuro. Desde que empecé el proyecto con el curso de *Kevin Werbach*, desarrollé un gran interés por la gamificación, tanto por mi pasión personal por el desarrollo software como por ser un usuario habitual de aplicaciones y juegos que utilizan estas técnicas.

6.2 Trabajo futuro

En cuanto al trabajo futuro, lo más evidente es la implementación de un prototipo de mi diseño propuesto para plataformas Android. Por otro lado, sería también interesante desarrollar la arquitectura para otros dispositivos como los que utilizan el sistema operativo iOS.

Una vez desarrollada, la arquitectura se puede completar con más funcionalidades de las propuestas, como por ejemplo mejorar la estructura que gestionará las redes sociales, para hacerla más completa, o también crear un sistema de almacenamiento de las características de los *leaderboards* para que se creen automáticamente en el servidor con un formato que estableció el programador y se envíen al cliente para que éste sólo tenga que mostrarlo por pantalla.

Capítulo 7: Planificación y Presupuesto

En este capítulo se mostrarán detalladamente las fases del proyecto así como la duración y presupuesto.

7.1 Planificación temporal

La planificación del proyecto se desglosa en la Tabla 3, destacando que los días que se han considerado son siempre laborables y se realizará posteriormente en el presupuesto una estimación de las horas trabajadas por día.

Concepto	Tiempo empleado	Inicio	Fin
Curso de gamificación de coursea	15 días	10/02/2014	28/02/2014
Estudio previo adicional	10 días	03/03/2014	13/03/2014
Planificación de la estructura proyecto	5 días	13/03/2014	17/03/2014
Requisitos técnicos	10 días	18/03/2014	30/03/2014

Estudio de las arquitecturas existentes	20 días	31/03/2014	24/04/2014
Propuestas de mejora de Userinfuser	7 días	24/04/2014	01/05/2014
Propuesta de arquitectura	20 días	05/05/2014	27/05/2014
Escritura de la memoria final	15 días	30/05/2014	17/06/2014
TOTAL	**102 DÍAS**	**10/02/2014**	**17/06/2014**

Tabla 3

7.2 Presupuesto

En cuanto al coste que tendría la realización de este proyecto de diseño se puede ver el desglose en la Tabla 4 [14].

Apellidos y nombre	Categoría	Dedicación (hombres mes)	Coste hombre mes	Coste (Euro)
Estévez Ayres, Iria Manuela	Ingeniero Senior	0,5	4.289,54	2.144,77
Ortiz Hernández, Daniel	Ingeniero Junior	2,35	2.694,39	6.331,82
	Hombres mes	**2,85**	**Total**	8.476,59

Tabla 4

Además en la tabla 5 se especifican los costes directos de los equipos.

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d)]

Asus K52F	1.200,00	100	3	60	60,00
				Total	60,00

Tabla 5

A los costes hay que sumarle el 20% de los costes indirectos, con lo que el presupuesto final quedaría como se detalla en la tabla 6.

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	8.477
Amortización	60
Costes Indirectos	1.707
Total	**10.244**

Tabla 6

Capítulo 8: Marco Legal

8.1 Introducción

Para el desarrollo del presente proyecto, ha de tenerse en cuenta el marco legislativo que puede afectar al desarrollo del mismo, para así ser consciente de las restricciones legales que pueden encontrarse en el desarrollo del software propuesto.

8.2 Aspectos legales

Dada la naturaleza del proyecto, que pretende dar un servicio en forma de *framework* o complemento a una aplicación independiente, las leyes aplicables serán las que se aplican a todo software desarrollado, entendiendo software por “un conjunto de instrucciones organizadas lógicamente (pues siguen una secuencia) y codificadas (porque se utiliza un lenguaje de programación para su desarrollo) y que tienen como fin resolver un problema o situación específica del usuario” [9].

8.2.1 Derechos de propiedad intelectual

Dado que el presente proyecto es un diseño creado a partir del esfuerzo intelectual, su desarrollo será una obra digital y por tanto estará sujeta a las leyes de propiedad intelectual. En este caso, el diseño de la arquitectura está

pensado para ofrecerse como solución comercial y por tanto sería necesaria la protección de su autoría original.[10]

8.2.2 Legislación aplicable a las aplicaciones móviles

El diseño se presenta como un complemento para aplicaciones móviles y por tanto deberá cumplir la legislación aplicable a las mismas. En concreto, estará sujeto a las siguientes normas [11]:

- Las funcionalidades de la aplicación deben ser lícitas.
- Se deben disponer de las licencias oportunas para todos los componentes incluidos en la aplicación.
- Hay que respetar los derechos de imagen y protección de datos. Se deben requerir los mínimos datos personales del usuario y dotarle a éste de la capacidad de modificación o borrado en cualquier momento.
- Para utilizar el software será necesario leer y aceptar la licencia de uso del mismo, para eximir tanto a desarrollador como cliente de tantas responsabilidades como sea posible.
- Dado que el *framework* propuesto podrá acceder a la memoria interna del dispositivo móvil, el usuario debe ser informado con antelación de este aspecto para que dé su visto bueno.

Además, las aplicaciones que se desarrollen con esta arquitectura deberán cumplir las normas que se establecen en el "acuerdo de distribución para desarrolladores" de *Android*[23].

[23] https://play.google.com/intl/ALL_es/about/developer-distribution-agreement.html

Capítulo 9: Bibliografía

1. *Gamificacion* (Fecha de consulta: 03 de 03 de 2014). Obtenido de http://www.gamificacion.com/

2. Europapress (Fecha de consulta: 5 de 03 de 2014). Obtenido de http://www.europapress.es/portaltic/internet/noticia-cityville-100-millones-usuarios-mensuales-20110115080001.html

3. Statisticbrain.com (Fecha de consulta: 5 de 03 de 2014) Obtenido de http://www.statisticbrain.com/call-of-duty-franchise-game-sales-statistics/

4. Werbach, K. (Fecha de consulta: 10 de 02 de 2014). Gamification. Philadelphia, Pensilvania, United States. https://www.coursera.org/course/gamification

5. Elconfidencial.com (Fecha de consulta: 7 de 03 de 2014) http://www.elconfidencial.com/tecnologia/2013-10-20/clash-of-clans-un-juego-gratuito-que-vale-3-000-millones-de-dolares_43576/

6. *Badgeville*. (Fecha de consulta: 10 de 04 de 2014). Obtenido de http://badgeville.com/solutions

7. Appscale (Fecha de consulta: 09 de 04 de 2014) Obtenido de http://www.appscale.com/

8. W3schools (Fecha de consulta: 09 de 04 de 2014) Obtenido de http://www.w3schools.com/tags/tag_iframe.asp

9. Dr. Aldo Elliot Segura (Fecha de consulta: 10 de 06 de 2014) Obtenido de http://www.derecho.usmp.edu.pe/cedetec/articulos/ASPECTOS%20JURIDICOS%20Y%20TECNICOS%20SOBRE%20EL%20SOFTWARE.pdf

10. Ministerio de Educación, Cultura y Deporte (Fecha de consulta: 10 de 06 de 2014) Obtenido de http://www.mcu.es/propiedadInt/CE/PropiedadIntelectual/Derechos.html

11. Iabspain (Fecha de consulta: 10 de 06 de 2014) Obtenido de http://www.iabspain.net/noticias/aspectos-legales-para-el-desarrollo-de-una-app/

12. Fortune (Fecha de consulta: 10 de 03 de 2014) Obtenido de http://fortune.com/2011/10/17/inside-the-gamification-gold-rush-2/

13. RAE (Fecha de consulta: 03 de 03 de 2014) Obtenido de http://www.rae.es/

14. Plantilla presupuesto UC3M (Fecha de consulta: 15 de 06 de 2014) https://www.uc3m.es/portal/page/portal/administracion_campus_leganes_est_cg/proyecto_fin_carrera/Formulario_PresupuestoPFC-TFG%20%283%29_1.xlsx

15. Koetsier, J. (Fecha de consulta: 06 de 02 de 2013). *VentureBeat.com*. Obtenido de http://venturebeat.com/2013/02/06/800-million-android-smartphones-300-million-iphones-in-active-use-by-december-2013-study-says/

16. V., J. (Fecha de consulta: 3 de Abril de 2013). *Codigoprogramacion*. Obtenido de http://codigoprogramacion.com/cursos/android/enviar-una-peticion-http-post-desde-android-a-aplicacion-web-php.html